BAB III
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Peran utama dalam mendukung pengembangan *website company profile* serta sistem internal untuk pengelolaan penawaran proyek di PT Artha Guna Sejati dimulai dengan briefing awal. Pada tahap ini, diberikan pemahaman mengenai struktur organisasi, proyek yang sedang berjalan, serta tugas dan tanggung jawab selama masa magang. Selain itu, terdapat pembekalan mengenai kebutuhan sistem yang akan dikembangkan.

Setelah itu, dilakukan analisis kebutuhan sistem dengan berdiskusi bersama *supervisor* dan tim terkait untuk memahami proses bisnis serta kendala dalam sistem yang berjalan saat ini. Tahap ini juga melibatkan pengumpulan data dari berbagai divisi terkait, seperti *form*at dokumen penawaran, alur *approval*, serta mekanisme transaksi yang digunakan perusahaan.

Setelah kebutuhan sistem teridentifikasi, tahap selanjutnya adalah perancangan *website* dan sistem internal. Pada tahap ini, dibuat rancangan awal berupa wireframe atau mockup untuk *website company profile* serta penyusunan struktur *database* yang akan digunakan dalam sistem internal. Sebelum pengembangan dimulai, rancangan tersebut harus melalui tahap review dan persetujuan dari *supervisor* serta tim terkait.

Tahap berikutnya adalah pengembangan *website* dan sistem internal dengan menggunakan HTML, CSS, dan JavaScript serta menghubungkanya dengan *database* DB_AGS. Pengembangan sistem internal mencakup fitur-fitur utama seperti *login*, transaksi penawaran, *approval* dokumen, serta daftar supplier. Selama proses ini, dilakukan uji coba secara berkala untuk memastikan sistem berjalan sesuai spesifikasi yang diharapkan. Sepanjang masa pengembangan, koordinasi dan evaluasi berkala dilakukan bersama *supervisor* dan pengguna sistem. Evaluasi ini bertujuan untuk mengidentifikasi kendala yang muncul serta melakukan perbaikan berdasarkan masukan dari tim perusahaan.

Setelah sistem dikembangkan, tahap selanjutnya adalah uji coba dan implementasi. Pengujian menyeluruh dilakukan terhadap sistem, mencakup aspek keamanan, per*forma*, dan kemudahan penggunaan. Selain itu, pelatihan juga diberikan kepada pengguna sistem agar dapat mengoperasikan sistem dengan optimal.

Tahap terakhir adalah presentasi akhir dan penyelesaian magang. Laporan akhir disusun dengan mencakup dokumentasi sistem, hasil pengembangan, serta evaluasi dari proyek yang telah dikerjakan. Hasil pengembangan kemudian dipresentasikan kepada tim perusahaan, beserta rekomendasi untuk pengembangan lebih lanjut. Setelah semua tanggung jawab terselesaikan, laporan akhir diserahkan sebelum penutupan program magang.

3.2 Tugas dan Uraian Kerja Magang

Selama periode kerja magang di PT Artha Guna Sejati, tanggung jawab utama adalah pengembangan *website company profile* serta sistem internal untuk pengelolaan penawaran proyek. Tujuan utama yang ingin dicapai adalah mengidentifikasi kebutuhan perusahaan dalam digitalisasi informasi dan proses penawaran proyek melalui analisis yang dilakukan bersama tim terkait. Selain itu, tujuan lainya adalah merancang *website* dan sistem internal yang sesuai dengan kebutuhan perusahaan.

Pengembangan dan implementasi sistem dilakukan menggunakan HTML, CSS, dan JavaScript, serta dihubungkan dengan *database* DB_AGS. Pengembangan ini mencakup fitur utama seperti *login*, transaksi penawaran, *approval* dokumen, dan daftar supplier. Setelah pengembangan sistem selesai, dilakukan uji coba untuk memastikan fungsionalitas berjalan dengan baik. Jika ditemukan kendala, perbaikan dilakukan berdasarkan hasil evaluasi dan masukan dari tim perusahaan. Sebagai langkah akhir, seluruh proses pengembangan didokumentasikan dalam laporan akhir yang berisi hasil kerja serta rekomendasi untuk pengembangan lebih lanjut. Laporan ini kemudian dipresentasikan kepada tim perusahaan guna

mendapatkan masukan serta memastikan bahwa sistem yang dikembangkan dapat digunakan secara optimal.

Tabel 2. 1 Timeline Kegiatan Magang

No.	Aktivitas	Minggu ke-	Tanggal Mulai Aktivitas	Tanggal Akhir Aktivitas
1.	Perkenalan perusahaan dan pembagian tugas awal	1 – 2	10 Februari 2025	23 Februari 2025
2.	Pembuatan *website company profile*	3 – 4	24 Februari 2025	7 Maret 2025
3.	Desain sistem internal *dashboard* (*head* & *admin*)	5 – 6	10 Maret 2025	21 Maret 2025
4.	Pembuatan halaman *head* (*upload* & *approval task*)	7	24 Maret 2025	27 Maret 2025
5.	Libur Magang	-	28 Maret 2025	6 April 2025
6.	Pengujian & validasi *form upload* (*head*)	8	7 April 2025	12 April 2025
7.	Pengembangan halaman *admin*	9 – 10	14 April 2025	25 April 2025
8.	Fitur revisi & melengkapi *task* (*admin mode*)	11 – 13	28 April 2025	17 Mei 2025
9.	Pengujian sistem Menyeluruh (*head-admin flow*)	14	19 Mei 2025	24 Mei 2025
10.	Bug fixing, UI refinement, *feedback*	15	26 Mei 2025	31 Mei 2025
11.	*Upload* ke hosting & integrasi *database*	16	2 Juni 2025	7 Juni 2025

12.	Pemantauan sistem & perbaikan kecil	17	9 Juni 2025	13 Juni 2025

Dari Tabel 2. ***1*** merupakan timeline kegiatan magang yang merupakan kegiatan selama magang sebagai *Web Developer* di PT Artha Guna Sejati berlangsung, memanfaatkan beberapa *tools* yang digunakan untuk menunjang proses kerja dan pengembangan *website*:

1. *Visual* Studio *Code*

   Visual Studio *Code* adalah *text* editor lintas *platform* yang menjadi alat utama dalam penulisan dan pengelolaan kode selama proses pengembangan. VS *Code* dipilih karena ringan, cepat, dan mendukung berbagai bahasa pemrograman serta ekstensi tambahan. Penulis menggunakan fitur seperti *Live Server* untuk melakukan live preview pada browser tanpa perlu refresh manual, serta fitur *IntelliSense* untuk auto-complete dan saran kode yang mempercepat penulisan logika JavaScript maupun struktur HTML. VS *Code* juga menyediakan terminal bawaan sehingga proses debugging dapat dilakukan langsung di satu tempat, tanpa berpindah-pindah antar aplikasi. Kelebihan lainya adalah kemampuan integrasi Git dan *file* explorer yang memudahkan navigasi dan pengorganisasian project yang cukup kompleks.
2. HTML, CSS, dan JavaScript

   Ketiga teknologi ini merupakan inti dari pengembangan website internal. HTML (Hyper*Text* Markup Language) digunakan untuk membangun struktur halaman web seperti *form input* data, daftar *task*, serta tombol navigasi. CSS (Cascading Style Sheets) digunakan untuk mengatur tampilan *visual* halaman, termasuk warna, tipografi, layout grid, responsivitas, dan elemen interaktif seperti hover dan animasi transisi. Sementara itu, JavaScript berperan sebagai bahasa pemrograman utama untuk mengatur seluruh logika dan alur sistem, mulai dari *upload file*, validasi data, pengelompokan *task* berdasarkan kategori, hingga interaksi

dengan penyimpanan lokal seperti IndexedDB dan localStorage. Kombinasi ketiganya membentuk satu sistem antarmuka pengguna yang dinamis dan berfungsi secara penuh tanpa perlu *backend* server.

3. Browser Developer Tools

   DevTools adalah fitur bawaan pada browser yang digunakan secara intensif untuk proses debugging dan optimasi *website*. Penulis menggunakan tab *Elements* untuk memeriksa dan memodifikasi struktur HTML secara real-time, tab *Console* untuk membaca log JavaScript dan menemukan error, serta tab *Application* untuk memantau isi IndexedDB dan localStorage. Selain itu, tab *Networ*k digunakan untuk melihat proses loading *file*, dan tab *Performance* digunakan untuk mengevaluasi kecepatan eksekusi kode dan animasi. DevTools membantu penulis mengidentifikasi masalah dengan cepat dan memastikan seluruh fitur berjalan optimal di sisi client.

4. cPanel dan *File Manager*

   Dalam pengembangan sistem *dashboard* internal perusahaan, peran *head* dan *admin* menjadi dua komponen penting yang bekerja saling melengkapi. *Head* berfungsi sebagai pihak pengunggah utama dokumen proyek, seperti BoQ, gambar teknis (DWG), spesifikasi produk, dan dokumen penawaran. Setiap dokumen yang diunggah oleh *head* akan membentuk satu tugas baru yang masuk ke sistem dan ditindaklanjuti oleh *admin*. *Admin* kemudian memeriksa, melengkapi, dan menyelesaikan tugas tersebut sebelum dikembalikan kepada *head* untuk disetujui atau direvisi. Seluruh proses ini dilakukan secara terstruktur dan terdokumentasi, dengan status yang terus diperbarui agar kedua pihak dapat memantau progres secara transparan.

   Agar sistem ini dapat berjalan dan diakses melalui jaringan internal perusahaan, seluruh *file* pendukung seperti HTML, CSS, dan JavaScript diunggah ke server menggunakan fitur *File Manager* dalam cPanel. cPanel sendiri merupakan sistem manajemen hosting berbasis web yang umum digunakan dan sangat mendukung kebutuhan pengembangan

aplikasi berbasis web. Dengan *File Manager*, pengelola sistem dapat mengatur struktur folder, mengedit *file* langsung dari browser, serta melakukan *upload file* proyek ke direktori public_html, sehingga halaman web dapat diakses oleh pengguna. Selain itu, cPanel juga menyediakan akses ke phpMy*Admin* yang memungkinkan pengelolaan *database* MySQL secara *visual* dan praktis, seperti memantau data tugas dari *head*, status *approval* oleh *admin*, serta catatan revisi yang tersimpan dalam tabel *database*. Kombinasi peran pengguna dan infrastruktur server ini membentuk sistem *dashboard* internal yang efisien, mudah diakses, dan mampu mendukung komunikasi dua arah secara digital dalam lingkungan kerja perusahaan.

Berdasarkan Tabel 2. ***1***, kegiatan harian selama magang di PT Artha Guna Sejati mencakup berbagai tugas yang berfokus pada pengembangan sistem berbasis web dan pengelolaan konten digital. Aktivitas dimulai dengan proses observasi kebutuhan sistem dan pengenalan alur kerja internal perusahaan, kemudian dilanjutkan dengan perancangan antarmuka, pembuatan halaman *website company profile*, serta pengembangan *dashboard* internal untuk manajemen dokumen penawaran proyek. Selama proses tersebut, dilakukan pengujian dan penyempurnaan fitur seperti *login*, unggah dokumen, *approval task*, serta tampilan katalog produk dan portofolio. Seluruh kegiatan ini dilakukan secara bertahap dan terdokumentasi dalam tabel harian sebagai bentuk pemantauan progres dan pencapaian selama masa magang.

3.2.1 Perkenalan perusahaan dan pembagian tugas awal.

Pada minggu pertama pelaksanaan kegiatan magang, dilakukan kegiatan orientasi sebagai tahap awal untuk memahami lingkungan kerja di PT Artha Guna Sejati. Tahapan ini mencakup berbagai aspek, mulai dari pengenalan struktur organisasi, penjabaran proses bisnis, hingga pemahaman mengenai peran divisi-divisi yang terlibat dalam operasional perusahaan. Hal ini bertujuan agar setiap kegiatan pengembangan yang dilakukan dapat terintegrasi dengan baik ke dalam sistem kerja yang sudah berjalan.

Struktur organisasi perusahaan dijelaskan secara menyeluruh, termasuk hubungan koordinatif antarbagian seperti divisi proyek, pengadaan, keuangan, serta administrasi. Pemahaman terhadap struktur ini mempermudah dalam menentukan alur informasi yang akan diakomodasi dalam sistem, khususnya dalam pengembangan fitur-fitur internal perusahaan seperti pengelolaan dokumen penawaran proyek dan manajemen tugas.

Selain struktur organisasi, informasi mengenai latar belakang, visi dan misi perusahaan juga disampaikan dengan lengkap. PT Artha Guna Sejati merupakan perusahaan yang bergerak di bidang konstruksi dan pengadaan barang, dengan fokus pada penyediaan solusi teknik dan logistik untuk berbagai proyek skala kecil, menengah, hingga besar. Visi perusahaan adalah menjadi mitra terpercaya dalam industri konstruksi, dan misi nya mencakup komitmen terhadap kualitas, efisiensi operasional, dan kepuasan klien. Informasi ini menjadi landasan dalam merancang tampilan dan konten *website* agar mampu mempresentasikan citra perusahaan secara profesional.

Di minggu yang sama juga dijelaskan secara umum mengenai proyek utama yang akan dikerjakan selama masa magang, yaitu pengembangan *website company profile* dari PT Artha Guna Sejati. *Website* ini

dirancang untuk menjadi representasi digital perusahaan dengan beberapa tujuan utama :

1. Menyediakan informasi profil perusahaan kepada publik secara ringkas namun in*form*atif
2. Menampilkan portofolio proyek sebagai bukti kapabilitas perusahaan dalam menangani berbagai jenis pekerjaan
3. Menjadi media komunikasi eksternal melalui halaman kontak dan permintaan informasi dari calon klien atau mitra
4. Menyediakan sistem internal bagi staf perusahaan untuk mengunggah dan memantau dokumen penawaran, revisi tugas, serta proses *approval* yang berkaitan dengan proyek.

Perencanaan sistem dilakukan berdasarkan kebutuhan perusahaan dan masukan dari pihak terkait. Dalam proses ini, perancangan antarmuka UI/UX juga menjadi perhatian utama agar tampilan *website* tidak hanya menarik secara *visual* tetapi juga fungsional dan mudah diakses. Pengenalan yang dilakukan secara menyeluruh pada minggu pertama ini menjadi fondasi penting sebelum memasuki tahap perancangan dan implementasi sistem. Dengan memahami struktur organisasi, *tools* yang digunakan, serta tujuan dari proyek pengembangan, proses kerja dapat dilaksanakan secara sistematis, efisien, dan sesuai dengan ekspektasi perusahaan.

3.2.2 Pembuatan *website company profile*.

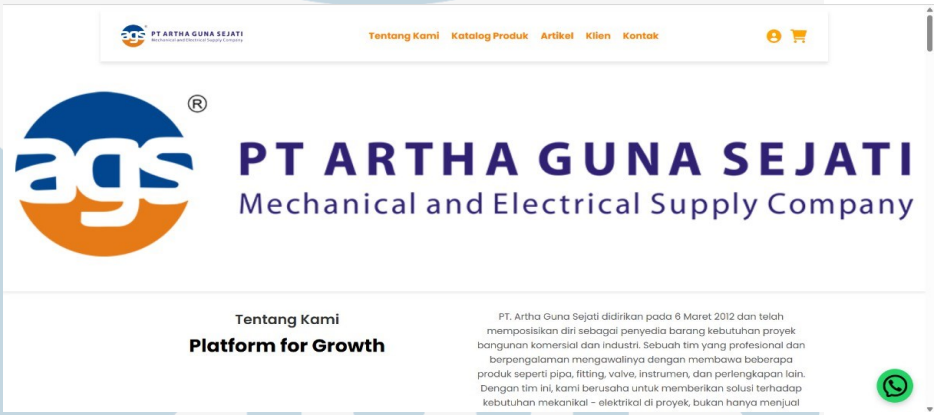
Perencanaan *layout* dasar, navigasi, dan antarmuka pengguna (*User Interface*) menjadi tahap awal dalam pengembangan *website company profile* PT Artha Guna Sejati. Langkah ini berfungsi sebagai fondasi *visual* dan struktural dari keseluruhan sistem yang akan dibangun, dengan tujuan agar pengalaman pengguna dapat berjalan secara intuitif dan in*form*atif.

Secara umum, *company profile* merupakan media yang menyajikan gambaran umum informasi mengenai identitas sebuah perusahaan. Informasi yang diberikan biasanya mencakup sejarah perusahaan, visi dan misi, layanan atau produk yang ditawarkan, struktur organisasi, pencapaian, hinggan informasi kontak. Tujuan utama dari *company profile* adalah untuk memperkenalkan perusahaan secara profesional kepada pihak luar, seperti calon klien, mitra bisnis, investor, maupun masyarakat umum. Seiring dengan perkembangan teknologi informasi dan digitalisasi bisnis, *company profile* tidak lagi hanya dibuat dalam bentuk dokumen cetak, melainkan dikembangkan menjadi sebuah *website* yang dapat diakses secara luas oleh publik. Keberadaan *website company profile* memiliki peran penting sebagai media representasi resmi perusahaan di dunia digital. Melalui *platform* digital ini, perusahaan dapat menunjukkan identitas dan meningkatkan kredibilitasnya secara lebih luas, karena kehadiran secara online menunjukkan profesionalisme dan kesiapan dalam menjangkau pasar yang lebih luas.

Website ini juga berperan sebagai wajah digital perusahaan yang dapat diakses kapan pun dan dimana pun. Keberadaan *website* ini menjadi bagian yang tidak terpisahkan dari strategi komunikasi dean branding perusahaan. Melalui *platform* digital ini, perusahaan dapat menunjukkan identitas dan kredibilitasnya secara lebih luas, profesional dan *real-time.* Dalam proses perancanganya, aspek desain antarmuka pengguna dan alur navigasi menjadi perhatian utama. Desain yang baik harus mampu menyampaikan pesan perusahaan secara *visual* dan fungsional. Navigasi yang jelas dan konsisten, tampilan yang responsif di berbagai perangkat, serta penggunaan elemen desain yang selaras dengan citra perusahaan adalah hal-hal yang dirancang secara cermat. Semua elemen ini bertujuan untuk memastikan bahwa publik

dapat memperoleh informasi yang dibutuhkan secara mudah, cepat, dan menyenangkan.

Dengan begitu, pengembangan *website company profile* PT Artha Guna Sejati tidak hanya difokuskan pada aspek teknis pembangunan sistem, tetapi juga mempertimbangkan bagaimana citra perusahaan direpresentasikan secara digital kepada publik. Tahap perencanaan layout dan antarmuka ini menjadi fondasi penting yang menentukan efektivitas dan profesionalisme tampilan akhir dari *website* perusahaan. Tahap implementasi desain *website* menjadi salah satu bagian paling penting dalam pengembangan sistem *company profile* PT Artha Guna Sejati. Setelah melalui proses perencanaan layout, struktur navigasi, serta penyusunan konten, langkah selanjutnya adalah menerjemahkan rancangan *visual* tersebut ke dalam sebuah *website* yang berfungsi secara penuh. Tujuanya adalah untuk menyampaikan informasi perusahaan secara profesional, in*form*atif, dan mudah diakses oleh publik, baik melalui perangkat komputer maupun ponsel.



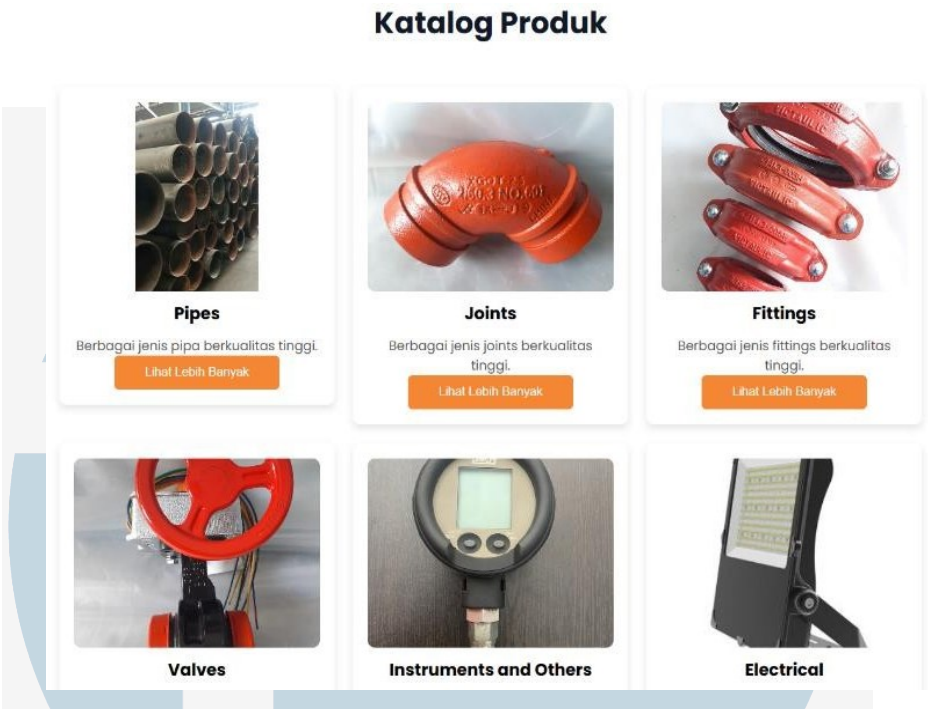
Gambar 3. 1 Halaman "Tentang Kami"

Website company profile PT Arta Guna Sejati dirancang untuk menampilkan dan menyampaikan informasi perusahaan, mulai dari profil perusahaan, katalog produk, portofolio proyek, hingga informasi kontak. Elemen-elemen tersebut diintegrasikan secara profesional

dalam satu *platform* digital yang memiliki navigasi intuitif dan tampilan antarmuka yang bersih dan responsif.

Berdasarkan Gambar 3. 1 yang menunjukkan halaman utama *website* yang menyajikan tampilan awal yang merepresentasikan identitas perusahaan. Pada bagian atas halaman terdapat *head*er yang berisi logo perusahaan dengan warna dominan biru dan oranye, serta nama perusahaan, serta *tagline* perusahaan "*Mechanical and Electrical Supply Company*" juga tercantum untuk memberikan gambaran singkat mengenai bidang usaha yang dijalankan. Elemen ini diletakkan secara strategis untuk membentuk kesan profesiobnal sejak pertama kali pengunjung mengakses situs. Pada gambar juga terdapat navigasi-navigasi yang dapat diakses dengan tujuan memudahkan pengunjung untuk menjelajahi berbagai informasi penting mengenai perusahaan.

Salah satu bagian penting dari halaman utama adalah segmen "Tentang Kami" yang memberikan penjelasan mengenai latar belakang PT Artha Guna Sejati. Penjelasan tersebut mencakup tanggal berdiri perusahaan, yaitu 6 Maret 2012, serta fokus utama perusahaan sebagai penyedia kebutuhan proyek mekanikal dan elektrikal untuk bangunan komersial dan industri. Penjelasan ini juga mencakup keunggulan perusahaan dalam menghadirkan solusi yang didukung oleh tim profesional dan produk berkualitas. Produk-produk tersebut meliputi pipa, fitting, valvel, instrument, dan perlengkapan lainya yang dibutuhkan untuk instalasi sistem mekanikal dan elektrikal. Penjelasan ini menekankan bahwa perusahaan tidak hanya berfokus pada penjualan, tetapi juga berkomitmen memberikan solusi untuk kebutuhan proyek mekanikal dan elektrikal.



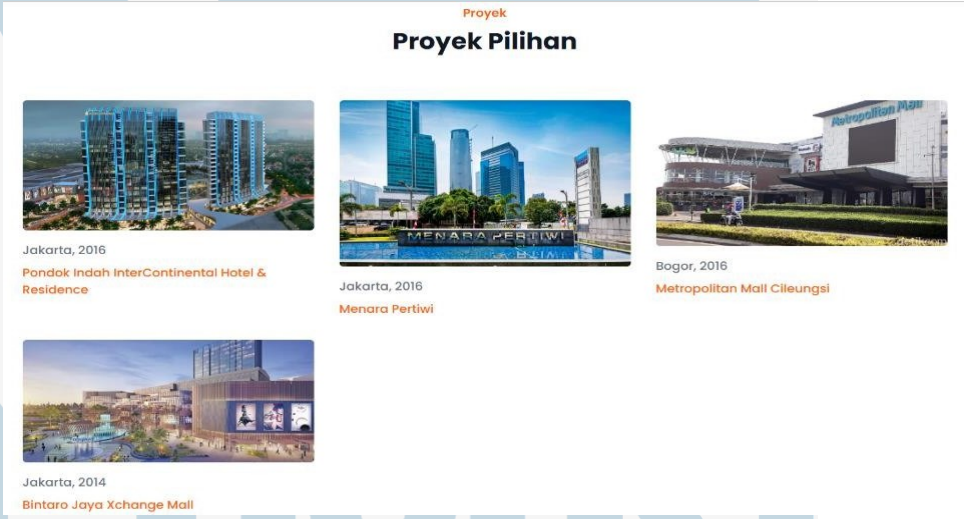
Gambar 3. 2 Halaman "Katalog Produk"

Pada Gambar 3. 2 yang menunjukkan Katalog Produk dimana pengunjung dapat menemukan berbagai kategori produk yang disediakan PT Artha Guna Sejati. Setiap produk dari produk yang dijual, ditampilkan dalam bentuk kotak informasi yang dilengkapi dengan gambar *representative*, nama kategori, deskripsi singkat, serta tombol "Lihat Lebih Banyak" untuk menampilkan detail produk lebih lanjut. Adapun kategori produk yang ditampilkan meliputi:

1. *Pipes*: Menyediakan berbagai macam dan jenis pipa dengan kualitas tinggi untuk kebutuhan mekanikal dan elektrikal dalam proyek-proyek industri dan komersial.
2. *Joints*: Menampilkan berbagai jenis sambungan pipa dengan kualitas tinggi yang berfungsi untuk menyambungkan bagian-bagian pipa sistem instalasi.
3. *Fittings*: Berisi komponen sambungan tambahan seperti *elbow, reducer,* dan *tee* yang mendukung integrasi sistem pipa sesuai spesifikasi proyek.

4. *Valves*: Menyediakan katup dengan berbagai macam tipe untuk mengatur aliran fluida pada sistem perpipaan,
5. *Instruments and Others*: Meliputi alat ukur atau instrumen pendukung serta komponen tambahan lainya yang diperlukan untuk instalasi sistem mekanikal dan elektrikal.
6. *Electrical*: Menyediakan perlengkapan elektrikal seperti lampu industri dan komponen kelistrikan lainya yang mendukung kebutuhan instalasi proyek.

Tampilan katalog ini dirancang tidak hanya untuk memperkenalkan produk, tetapi juga untuk memberikan kenyamanan *visual* dan kemudahan akses informasi bagi calon klien atau pengunjung yang ingin mengetahui lebih lanjut mengenai penawaran perusahaan.



Gambar 3. 3 Halaman "Proyek Pilihan"

Pada Gambar 3. 3 ditampilkan beberapa proyek besar yang pernah ditangani oleh PT Artha Guna Sejati sebagai bentuk portofolio perusahaan yang memberikan nilai tambah signifikan bagi citra perusahaan. Melalui halaman ini, ditampilkan dokumentasi berupa proyek besar yang telah berhasil diselesaikan oleh PT Artha Guna Sejati. Informasi pada setiap proyek meliputi nama proyek, lokasi, tahun pengerjaan, dan dokumentasi *visual* berupa foto. Beberapa proyek yang ditampilkan antara lain Pondok Indah InterContinental

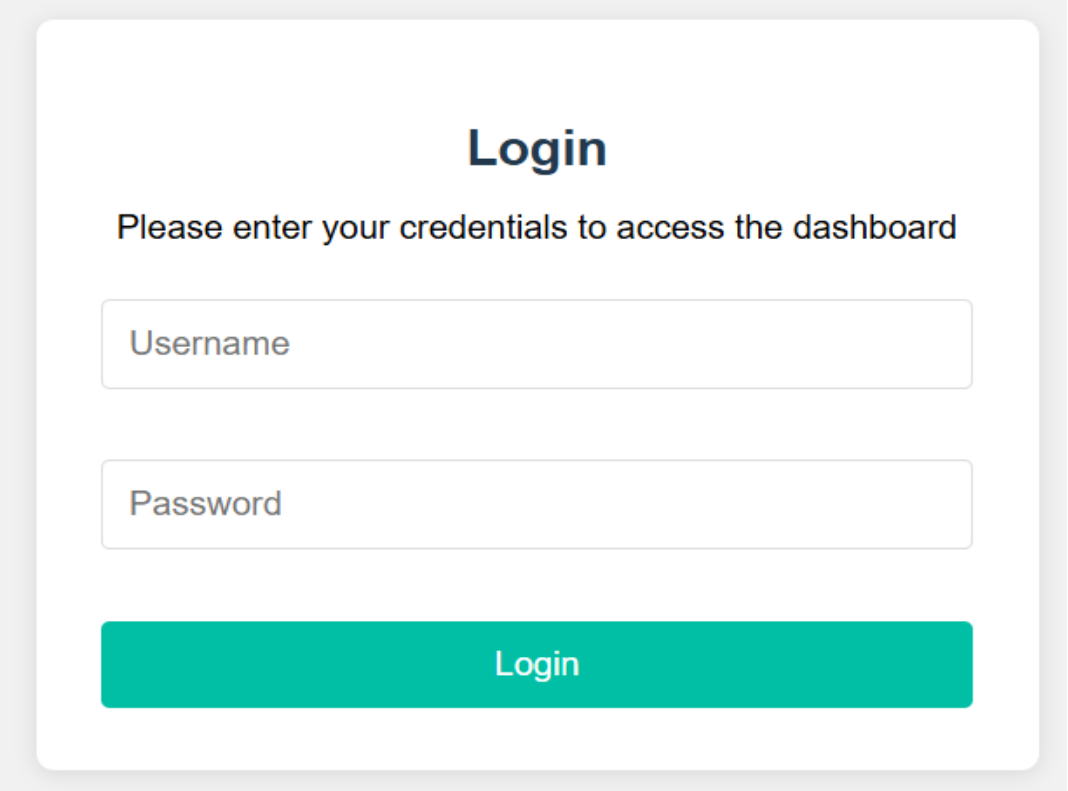
Hotel & Residence (Jakarta, 2016), Menara Pertiwi (Jakarta, 2016), Metropolitan Mall Cileungsi (Bogor, 2016), dan Bintaro Jaya Xchange Mall (Jakarta, 2014). Penyusunan halaman ini tidak hanya menunjukkan pengalaman dan keberhasilan perusahaan dalam menangani proyek berskala besar, tetapi juga menjadi alat promosi *visual* yang efektif untuk membangun kepercayaan pengunjung terhadap kapabilitas dan kredibilitas perusahaan.

Secara keseluruhan, implementasi desain ke dalam bentuk *website* fungsional telah dilakukan dengan pendekatan yang terstruktur dan berorientasi pada pengguna. Setiap halaman dan elemen *websit*e dirancang untuk menyampaikan identitas perusahaan secara profesional sekaliguas memberikan pengalaman akses informasi yang nyaman. *Website* ini menjadi representasi digital yang mampu menunjang kebutuhan komunikasi, promosi, dan transparansi informasi perusahaan secara menyeluruh.

3.2.3 Desain sistem internal *dashboard* (*head* & *admin*).

Pada tahap awal pengembangan sistem internal PT Artha Guna Sejati, dilakukan perancangan awal tata letak (*layout*) dan alur kerja (*workflow*) dari sebuah *platform* berbasis web yang dirancang khusus untuk mendukung operasional internal perusahaan, terutama dalam hal pengelolaan dokumen proyek. Sistem ini dikembangkan sebagai respons terhadap kebutuhan perusahaan untuk memiliki media penyimpanan dokumen yang terpusat, tertata, dan mudah diakses oleh pihak internal, tanpa harus menggunakan metode manual atau berpindah-pindah *platform*.

Langkah pertama dalam pengembangan ini adalah membuat halaman *login* statis sebagai pintu masuk utama ke dalam sistem. Pada tahap ini, proses autentikasi pengguna masih bersifat sederhana, yaitu dengan menggunakan data *username* dan *password* yang telah ditentukan secara statis di dalam kode program (*hardcoded*).



Gambar 3. 4 Halaman Login

Setelah pengguna berhasil melakukan *login*, sistem akan mengarahkan mereka ke halaman *dashboard* utama yang disesuaikan dengan peran masing-masing, yakni *Head*, *Manager* dan *Admin*. Hal ini dilakukan agar setiap pengguna memiliki pengalaman yang fokus serta tugas yang jelas sesuai dengan tanggung jawabnya. Berikut merupakan *code* yang terdapat pada halaman *login* ini :

```css
/* Role-specific menu visibility */
.admin-only {
  display: none; /* Hidden by default, shown for admin role */
}
.super-admin-only {
  display: none; /* Hidden by default, shown for manager role */
}
.super-admin-only-not-full {
  display: none; /* Hidden by default, shown only for manager but not for head */
}
.full-admin-only {
  display: none; /* Hidden by default, shown for head role */
}
```

Gambar 3. 5 Kode CSS User

Gambar 3. 5 menunjukkan potongan kode CSS yang ditampilkan pada gambar pertama merupakan elemen penting dalam sistem manajemen tampilan antarmuka pengguna (UI) yang dirancang untuk disesuaikan dengan peran atau role pengguna dalam sistem. Dalam pengembangan ini, struktur peran dibedakan menjadi beberapa kategori, seperti *admin*, *super admin*, dan *full admin* (*head*), di mana masing-masing peran memiliki akses yang berbeda terhadap fitur dan tampilan tertentu.

Setiap *class* CSS, seperti *.admin-only*, *.super-admin-only*, *.super-admin-only-not-full*, dan *.full-admin-only*, secara *default* disetel dengan *display: none*;. Ini berarti bahwa elemen HTML yang memiliki *class* tersebut tidak akan terlihat saat halaman pertama kali dimuat. Namun, melalui penggunaan JavaScript atau logika *backend*, *class- class* ini dapat diaktifkan, sehingga visibilitas elemen dapat diubah sesuai dengan hak akses pengguna yang sedang *login*.

Sebagai contoh, elemen dengan *class .admin-only* hanya akan ditampilkan kepada pengguna yang terautentikasi sebagai *admin*. Sementara itu, elemen dengan *class .super-admin-only* hanya akan terlihat bagi pengguna yang memiliki peran sebagai *manager* atau *super admin*. *Class .super-admin-only-not-full* dirancang untuk ditampilkan

kepada *manager*, tetapi tidak untuk peran *head*, sedangkan *class .full-admin-only* hanya akan terlihat oleh pengguna yang menjabat sebagai kepala divisi (*head*).

Konfigurasi ini tidak hanya membuat antarmuka sistem menjadi lebih dinamis, tetapi juga meningkatkan keamanan. Dengan cara ini, fitur-fitur tertentu hanya akan muncul bagi role yang berwenang mengaksesnya, sehingga mengurangi risiko kesalahan akses atau manipulasi data oleh pihak yang tidak berwenang.

```
<!-- Login Container -->
<div id="loginContainer" class="login-container">
  <div class="login-header">
    <h2>Login</h2>
    <p>Please enter your credentials to access the dashboard</p>
  </div>
  <div id="loginError" class="login-error" style="display: none;"></div>
  <form id="loginForm" class="login-form">
    <div class="form-group">
      <input type="text" id="username" name="username" placeholder="Username" required />
    </div>
    <div class="form-group">
      <input type="password" id="password" name="password" placeholder="Password" required />
    </div>
    <button type="submit" class="login-button">Login</button>
  </form>
</div>
```

Gambar 3. 6 Kode HTML Struktur Login

Gambar 3. 6 menampilkan kode HTML dari struktur *form login* yang digunakan untuk proses autentikasi pengguna sebelum mereka dapat mengakses sistem *dashboard*. *Form login* ini merupakan komponen krusial dalam sistem keamanan, berfungsi sebagai pintu masuk utama menuju seluruh fitur aplikasi.

Form ini dibungkus dalam sebuah *container* dengan id="*login*-container", yang mengatur tampilan keseluruhan *area login*. Di dalamnya terdapat bagian judul (<h2>*Login*</h2>) dan deskripsi singkat yang memberikan arahan kepada pengguna untuk mengisi kredensial mereka.

Komponen utama dari *form login* terdiri dari dua *input*:

- *Username*: Tipe *input* ini adalah *text* dengan atribut *required*, yang berarti pengguna wajib mengisi kolom ini untuk melanjutkan.
- *Password*: Tipe *input* ini adalah *password*, juga dengan atribut *required*, untuk menjaga kerahasiaan kata sandi pengguna.

Selain itu, terdapat tombol <button type="submit"> yang berfungsi untuk mengirimkan *form* dan mencoba melakukan *login*.

Elemen disiapkan untuk menampilkan pesan kesalahan jika proses *login* gagal, misalnya ketika *username* atau *password* yang dimasukkan tidak cocok. Pada awalnya, div ini disembunyikan dengan properti style="display: none;" dan akan ditampilkan secara dinamis melalui JavaScript jika terjadi kesalahan.

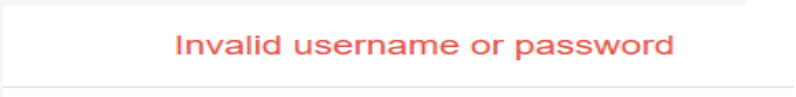
```
// Validate required parameters
if (!isset($data['username']) || !isset($data['password'])) {
    echo json_encode(['success' => false, 'message' => 'Missing required parameters: username and password']);
    exit;
}
```

Gambar 3. 7 Kode PHP Validasi Login

Form login ini merupakan bagian dari pendekatan dasar dalam proses autentikasi. Integrasi *form* ini dengan *backend* (misalnya menggunakan PHP) akan memeriksa kecocokan data yang di*input* dengan *database* pengguna. Jika *login* berhasil, pengguna akan diarahkan ke halaman yang sesuai berdasarkan *role*-nya, sehingga memastikan pengalaman pengguna yang aman dan terstruktur.

Pada Gambar 3. 7 ditampilkan potongan kode PHP yang digunakan untuk menangani proses *login* pengguna pada sistem internal perusahaan. Kode ini merupakan bagian inti dari validasi server-*side*, khususnya dalam memverifikasi *input username* dan *password* yang diberikan oleh pengguna sebelum mengakses *dashboard* sesuai peran masing-masing. Validasi ini sangat penting dilakukan untuk menjaga keamanan data dan memastikan bahwa hanya pengguna yang terdaftar dan memiliki hak akses sah yang dapat masuk ke dalam sistem.

Melalui kode tersebut, sistem pertama-tama akan memeriksa apakah *request* yang diterima merupakan metode POST dan kemudian mengecek keberadaan parameter *username* dan *password*. Apabila salah satu parameter tersebut tidak dikirimkan, maka sistem akan langsung mengembalikan respons berupa pesan error dengan notifikasi bahwa terdapat parameter yang belum diisi, dalam hal ini "Missing required parameters: *username* and *password*". Hal ini dilakukan untuk memastikan bahwa *input* yang diberikan lengkap sebelum dilanjutkan ke tahap validasi berikutnya.



Gambar 3. 8 Notifikasi Peringatan Kesalahan Login

Setelah data diterima dengan lengkap, sistem akan melakukan pencocokan *username* dengan data yang ada pada tabel *admin*s di *database*. Apabila sistem tidak menemukan *username* yang cocok, atau jika *password* yang diberikan tidak sesuai setelah diverifikasi menggunakan fungsi *password_verify*(), maka sistem akan menampilkan peringatan berupa teks "*Invalid username* or *password*". Peringatan ini ditampilkan dalam *format* JSON dan nantinya dapat ditangkap oleh antarmuka pengguna untuk disampaikan secara *visual* kepada pengguna agar segera melakukan koreksi.

Fitur peringatan seperti ini sangat berguna dalam membantu pengguna mengenali kesalahan *input*, tanpa mengungkapkan rincian teknis atau struktur data di balik sistem. Penanganan error yang in*form*atif namun tetap aman seperti ini berkontribusi terhadap pengalaman pengguna yang baik (*user-friendly*) sekaligus menjaga aspek keamanan dari sisi *backend*.

Dengan demikian, proses validasi *login* ini tidak hanya bertugas menyaring *input* yang sah secara teknis, tetapi juga mengarahkan pengguna untuk mengoreksi kesalahan secara mandiri sebelum dapat melanjutkan akses ke dalam sistem. Penerapan validasi dan umpan balik yang tepat menjadi kunci dalam menciptakan sistem *login* yang efisien, aman, dan ramah pengguna.

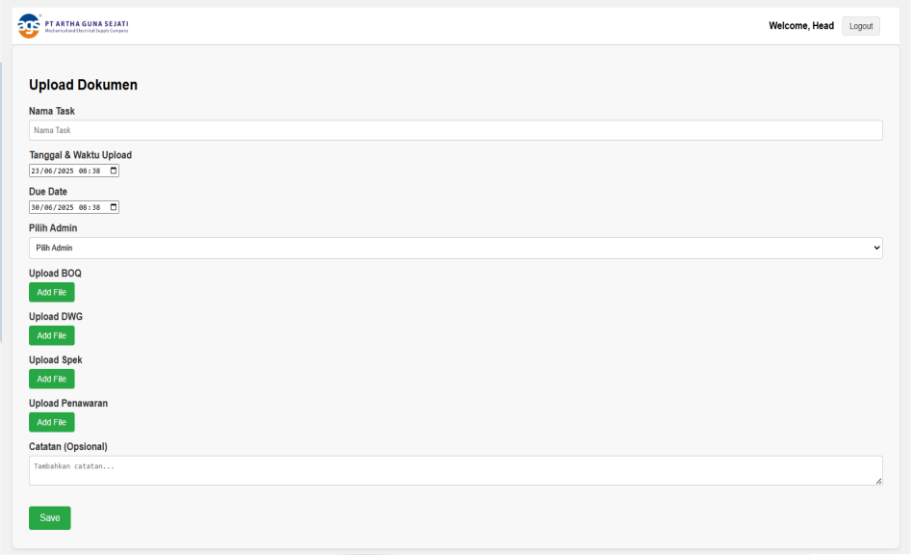
3.2.4 Pembuatan halaman *head* dan *manager* (*upload*, *history* & *approval task*).

Dalam pengembangan sistem internal perusahaan, halaman ***head & manager*** memegang peranan penting sebagai pusat pengelolaan awal untuk semua dokumen dan tugas yang akan ditangani oleh tim *admin*. Halaman ini secara khusus didesain agar kepala divisi atau penanggung jawab utama bisa mengunggah berbagai dokumen penting seperti **Bill** *of Quantity* (BoQ), gambar teknis (DWG), spesifikasi produk, dan dokumen penawaran lainya. Selain fungsi *upload*, halaman ini juga dilengkapi dengan fitur *approval task*, di mana pengguna dengan peran *head* dapat melakukan pengecekan terhadap tugas yang sudah diunggah, memberikan persetujuan, serta memberikan umpan balik atau meminta revisi apabila diperlukan.

Pembuatan halaman ini bertujuan untuk menciptakan alur kerja yang efisien dan terdokumentasi dengan baik. Dengan begitu, dokumen kerja tersimpan secara terstruktur dan risiko kehilangan data atau

miskomunikasi dalam pelaksanaan proyek internal dapat diminimalkan.

Proses pengembangan dimulai dengan perancangan struktur dasar halaman menggunakan HTML. Struktur ini mencakup *form input* untuk nama tugas, unggahan *file* BoQ, *file* DWG, spesifikasi produk, dokumen penawaran, serta kolom catatan tambahan untuk keperluan penjelasan atau instruksi dari *user head.*



Gambar 3. 9 Halaman "Upload Dokumen"

Selanjutnya, fitur *upload* dokumen diimplementasikan dengan memanfaatkan HTML dan JavaScript untuk bagian antarmuka, kemudian diproses melalui *backend* PHP agar data *file* dan metadata terkait dapat tersimpan dengan aman di *database* perusahaan. Dokumen yang diunggah akan disimpan secara fisik di direktori server, sementara informasi penting seperti nama *file*, nama tugas, tanggal *upload*, dan status awal tugas akan tercatat dalam tabel *tasks*.

```
<form id="uploadForm" enctype="multipart/form-data" method="POST" action
    ="uploadFile.php">
    <div id="formData" class="form-container" style="display:block;">
      <h2>Upload Dokumen</h2>
      <div class="form-group">
        <label>Nama Task</label>
        <input type="text" id="taskName" name="taskName" placeholder="Nama
            Task" required />
      </div>
      <div class="form-group">
        <label>Tanggal & Waktu Upload</label>
        <input type="datetime-local" id="tanggalWaktu" name="tanggalWaktu"
            />
      </div>
      <div class="form-group">
        <label>Due Date</label>
        <input type="datetime-local" id="dueDate" name="dueDate" required
            />
      </div>
      <div class="form-group">
        <label>Pilih Admin</label>
        <select id="adminSelect" name="adminId" required>
          <option value="">Pilih Admin</option>
        </select>
      </div>
      <div class="form-group">
        <label>Upload BOQ</label>
        <div class="file-upload-container">
          <input type="file" id="boqFile" name="boqFile[]" accept=".pdf
              ,.xls,.xlsx,.jpg,.png,.zip,.dwg" style="display: none;" />
          <button type="button" class="add-file-btn" onclick="addMoreFiles
              ('boqFile')">Add File</button>
        </div>
        <div id="boqFileList" class="file-list"></div>
      </div>
      <div class="form-group">
        <label>Upload DWG</label>
        <div class="file-upload-container">
          <input type="file" id="dwgFile" name="dwgFile[]" accept=".pdf
              ,.xls,.xlsx,.jpg,.png,.zip,.dwg" style="display: none;" />
          <button type="button" class="add-file-btn" onclick="addMoreFiles
              ('dwgFile')">Add File</button>
        </div>
        <div id="dwgFileList" class="file-list"></div>
      </div>
      <div class="form-group">
        <label>Upload Spek</label>
        <div class="file-upload-container">
          <input type="file" id="spekFile" name="spekFile[]" accept=".pdf
              ,.xls,.xlsx,.jpg,.png,.zip,.dwg" style="display: none;" />
          <button type="button" class="add-file-btn" onclick="addMoreFiles
              ('spekFile')">Add File</button>
        </div>
        <div id="spekFileList" class="file-list"></div>
      </div>
      <div class="form-group">
        <label>Upload Penawaran</label>
        <div class="file-upload-container">
          <input type="file" id="penawaranFile" name="penawaranFile[]"
              accept=".pdf,.xls,.xlsx,.jpg,.png,.zip,.dwg" style="display:
              none;" />
          <button type="button" class="add-file-btn" onclick="addMoreFiles
              ('penawaranFile')">Add File</button>
        </div>
        <div id="penawaranFileList" class="file-list"></div>
      </div>
      <div class="form-group">
        <label>Catatan (Opsional)</label>
        <textarea id="uploadNote" name="note" placeholder="Tambahkan
            catatan..."></textarea>
      </div>
      <button type="button" class="save-btn" onclick="submitTask()">Save
          </button>
    </div>
```

Gambar 3. 10 Kode HTML Halaman "Upload Dokumen"

Gambar 3. 10 merupakan kode HTML halaman "*Upload* Dokumen" yang menampilkan kode HTML yang ditampilkan dalam gambar tersebut merupakan bagian dari halaman *formulir* unggahan dokumen yang dirancang khusus untuk pengguna dengan peran kepala divisi (*head*). *Formulir* ini digunakan untuk memasukkan data terkait tugas atau pekerjaan yang akan dikelola oleh *admin*, lengkap dengan dokumen-dokumen pendukung yang perlu diunggah.

Formulir ini dimulai dengan judul "*Upload* Dokumen" dan mencakup beberapa elemen *input* yang penting. Pertama, terdapat kolom untuk mengisi nama tugas yang harus diisi. Selanjutnya, terdapat dua kolom *input* bertipe datetime-local, yang masing-masing digunakan untuk mencatat tanggal dan waktu unggah serta batas waktu (due date) dari tugas tersebut. *Form* ini juga menyediakan menu dropdown untuk memilih *admin* yang akan menangani tugas, dan kolom ini juga bersifat wajib. Pemilihan *admin* dilakukan melalui elemen <select>, sehingga pengguna dapat menentukan siapa yang akan bertanggung jawab untuk menindaklanjuti dokumen tersebut.

Selanjutnya, terdapat beberapa bagian yang mengatur proses unggah dokumen, seperti dokumen BoQ, gambar DWG, spesifikasi produk, dan dokumen penawaran. Setiap kategori *file* memiliki *input* bertipe *file* yang disembunyikan secara *visual*, dan digantikan oleh tombol "*Add File*" yang akan memicu fungsi JavaScript untuk membuka jendela pemilihan *file*. *File* yang diunggah dapat berupa berbagai *form*at sesuai dengan kebutuhan dokumen teknis proyek. Setiap kategori unggahan *file* juga dilengkapi dengan *area* bernama *file*-list, yaitu sebuah div yang digunakan untuk menampilkan nama *file* yang telah dipilih, sehingga pengguna dapat memverifikasi *file* sebelum mengirimnya. Di bagian akhir *formulir*, terdapat kolom catatan tambahan yang bersifat opsional. Kolom ini disediakan untuk memberi ruang bagi

pengguna dalam menambahkan instruksi atau informasi pendukung terkait tugas. Terakhir, tombol "Save" yang terletak di bagian bawah akan menjalankan fungsi JavaScript submit*Task*() saat diklik, bukan langsung mengirimkan *formulir* seperti pada tombol bertipe submit biasa. Hal ini memungkinkan proses validasi dilakukan terlebih dahulu sebelum *formulir* benar-benar dikirim ke *file backend uploadFile*.php.

Setelah pengguna mengisi seluruh data pada *form upload* tugas dan menekan tombol "Save", sistem akan menjalankan fungsi JavaScript yang sebelumnya sudah dikonfigurasikan untuk mengirimkan data *input* beserta *file-file* yang dipilih ke server melalui metode POST. Permintaan ini diarahkan ke *file* submit_*task*.php yang menangani seluruh proses penyimpanan data ke *database*.

Di dalam *file* submit_*task*.php, server menerima data yang dikirim melalui *form*, termasuk informasi teks seperti nama tugas, tanggal, dan catatan, serta *file* yang diunggah seperti BoQ, DWG, spek, dan penawaran. *File-file* tersebut diproses terlebih dahulu menggunakan fungsi *uploadFile*(), lalu hasil nama *file* yang sudah tersimpan di server disiapkan untuk disimpan ke dalam *database*. Bagian berikut merupakan inti dari proses penyimpanan data ke *database*:

```php
// Query insert dengan prepared statement agar aman dari SQL Injection
$sql = "INSERT INTO tasks (nama_task, tanggal_waktu, boq, dwg, spek, penawaran, note, status)
        VALUES (?, ?, ?, ?, ?, ?, ?, 'pending')";

$stmt = $conn->prepare($sql);
if (!$stmt) {
    echo json_encode(["status" => "error", "message" => "Gagal prepare statement: " . $conn->error]);
    exit;
}

$stmt->bind_param("sssssss", $namaTask, $tanggalWaktu, $boq, $dwg, $spek, $penawaran, $note);

if ($stmt->execute()) {
    echo json_encode(["status" => "success"]);
} else {
    echo json_encode(["status" => "error", "message" => $stmt->error]);
}

$stmt->close();
$conn->close();
```

Gambar 3. 11 Kode *JavaScript* Submit Task

Gambar 3. 11 merupakan kode *JavaScript Submit Task* yang menampilkan *JavaScript* dari submit_*task*.php yang menyusun perintah SQL untuk menyimpan data ke dalam tabel *tasks*. Data dikirim menggunakan teknik prepared statement agar aman dari serangan SQL Injection. Fungsi bind_param mengaitkan nilai dari setiap variabel yang dikirim dari *form* ke tempat yang sesuai pada *query* SQL. Setelah semua data terisi dan valid, sistem mengeksekusi *query* tersebut dan jika berhasil, sistem akan mengembalikan respons JSON dengan status success yang bisa ditangkap kembali oleh *JavaScript* di sisi klien untuk menampilkan *not*ifikasi keberhasilan atau memperbarui tampilan halaman. Namun, jika terjadi kesalahan saat penyimpanan, sistem akan mengembalikan pesan error yang menjelaskan masalahnya. Secara keseluruhan, proses ini memastikan bahwa setiap tugas yang dikirim oleh pengguna akan tersimpan secara otomatis ke dalam *database* ptarthag_pt_ags dengan status awal "pending", dan selanjutnya dapat dilihat serta diproses lebih lanjut oleh pengguna *admin*.

```php
// Function to handle file uploads
function processUploads($fileField) {
    global $uploadDir;
    $uploadedPaths = [];

    // Check if files were uploaded for this field
    if (!isset($_FILES[$fileField]) || empty($_FILES[$fileField]['name'][0]
        )) {
        return [];
    }

    // Process each uploaded file
    $fileCount = count($_FILES[$fileField]['name']);
    for ($i = 0; $i < $fileCount; $i++) {
        if ($_FILES[$fileField]['error'][$i] !== UPLOAD_ERR_OK) {
            error_log("Upload error for {$fileField}[$i]: " .
                $_FILES[$fileField]['error'][$i]);
            continue;
        }

        // Generate unique filename
        $fileName = time() . '_' . basename($_FILES[$fileField]['name'][$i]
            );
        $targetPath = $uploadDir . $fileName;

        // Move the uploaded file
        if (move_uploaded_file($_FILES[$fileField]['tmp_name'][$i],
            $targetPath)) {
            error_log("File uploaded successfully: $targetPath");
            $uploadedPaths[] = $targetPath;
        } else {
            error_log("Failed to move uploaded file: " .
                $_FILES[$fileField]['name'][$i]);
        }
    }

    return $uploadedPaths;
}
```

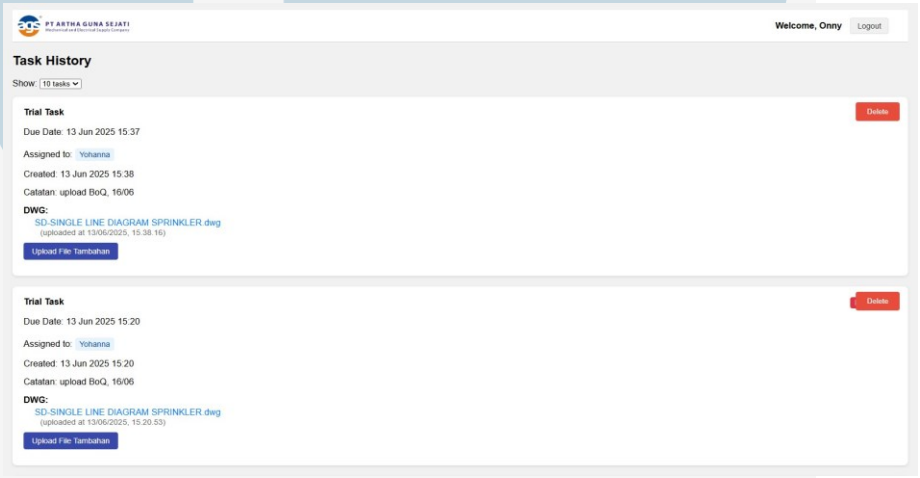
Gambar 3. 12 Kode *JavaScript* Upload File

Gambar 3. 12 merupakan kode *JavaScript Upload File* dimana menampilkan potongan *code uploadFile*.php di *JavaScript* dengan memperlihatkan fungsi `process*Upload*s($*file*Field)` merupakan komponen penting dalam sistem unggah *file* yang bertugas untuk mengelola proses penyimpanan *file* yang diunggah oleh pengguna. Fungsi ini menerima satu parameter yang merupakan nama field *input* dari *formulir* HTML yang digunakan untuk mengunggah *file*, seperti `boq*File*`, `dwg*File*`, dan lainya. Di awal fungsi, variabel global `$*uploadd*ir` diakses agar dapat digunakan

sebagai direktori tujuan untuk menyimpan *file* yang berhasil diunggah. Selanjutnya, dibuatlah array kosong bernama `$*upload*edPaths` yang akan menyimpan daftar path *file* yang berhasil diproses.

Langkah pertama dalam fungsi ini adalah memeriksa apakah *file* benar-benar ada dan apakah field yang bersangkutan berisi *file*. Jika tidak ada *file* yang diunggah, fungsi akan segera mengembalikan array kosong. Namun, jika *file* tersedia, sistem akan menghitung jumlah *file* yang diunggah dalam satu field menggunakan fungsi `count()`. Setelah itu, dilakukan perulangan untuk memproses setiap *file* satu per satu. Setiap *file* akan diperiksa apakah terjadi kesalahan saat proses unggah dengan membandingkan nilai `$_*FILE*S[$*file*Field]['error'][$i]` dengan `*UPLOAD*_ERR_OK`. Jika terdapat kesalahan, *file* tersebut akan dilewati dan tidak akan diproses lebih lanjut. Untuk setiap *file* yang valid, sistem akan membuat nama *file* baru yang unik dengan menambahkan timestamp (`time()`) pada nama *file* asli. Nama ini kemudian digabungkan dengan direktori tujuan untuk membentuk path lengkap penyimpanan *file* (`$targetPath`). Selanjutnya, *file* akan dipindahkan dari direktori sementara (`tmp_name'). Jika proses pemindahan berhasil, path *file* tersebut akan ditambahkan ke dalam array `$*upload*edPaths`. Di akhir fungsi, semua path *file* yang berhasil diunggah akan dikembalikan dalam bentuk array. Hasil ini nantinya dapat disimpan ke dalam *database* atau digunakan untuk keperluan lain dalam sistem. Dengan adanya fungsi ini, sistem dapat menangani banyak *file* sekaligus dari berbagai kategori dokumen dengan cara yang aman dan terstruktur. Sebagai pelengkap dari proses unggah, sistem juga menyediakan halaman *Task History* khusus pada antarmuka pengguna *manager*. Halaman ini digunakan untuk memantau seluruh riwayat tugas yang telah dibuat maupun yang sedang dalam proses. Informasi yang

ditampilkan meliputi nama tugas, tanggal unggah, status terbaru (seperti pending, revisi, approved, atau rejected), dan riwayat interaksi seperti revisi atau penambahan *file* oleh *admin*. Fitur ini mempermudah *manager* dalam melakukan pelacakan progres pekerjaan, mengawasi kinerja masing-masing tugas, serta mengambil keputusan berdasarkan histori tugas yang terdokumentasi dengan baik. Dengan demikian, *manager* dapat menjaga kelancaran alur kerja dan memastikan bahwa semua dokumen proyek telah diselesaikan sesuai waktu dan standar yang diharapkan.



Gambar 3. 13 Halaman "Task"

Gambar 3. 13 merupakan Halaman "*Task*" memperlihatkan tampilan halaman *Task History* yang diakses oleh pengguna dengan peran *manager* atau *head*. Pada halaman ini, pengguna dapat melihat daftar tugas yang sudah dibuat maupun yang sedang berjalan. Informasi yang ditampilkan meliputi nama tugas, tanggal jatuh tempo (*due date*), waktu pembuatan, penerima tugas (*assigned to*), catatan tambahan, dan *file* yang telah diunggah beserta keterangan waktu unggah. Terdapat pula tombol "*Upload File* Tambahan" yang memungkinkan *manager* atau *head* menambahkan

dokumen pelengkap apabila diperlukan. Tombol "Delete" juga tersedia untuk menghapus *task* yang tidak valid atau tidak diperlukan lagi.

Untuk mendukung tampilan dinamis ini, sistem menggunakan *JavaScript* untuk memuat dan menampilkan data tugas secara real-time dari *database*. Berikut adalah potongan kode *JavaScript* yang digunakan untuk mengelola dan menampilkan isi halaman *Task History*:

```
    // Get tasks created by this super_admin with admin information
    $sql = "SELECT t.id, t.nama_task, t.tanggal_upload, t.status, a.name AS
        admin_name,
                    t.created_at, t.catatan, t.boq, t.dwg, t.spek, t.penawaran, t
                        .tambahan
            FROM tasks t
            LEFT JOIN admins a ON t.admin_id = a.id
            WHERE t.creator_id = ?
            ORDER BY t.created_at DESC
            LIMIT ?";

    $stmt = $conn->prepare($sql);
    if (!$stmt) {
        throw new Exception("Prepare statement failed: " . $conn->error);
    }

    $stmt->bind_param("ii", $userId, $limit);
    $stmt->execute();
    $result = $stmt->get_result();

    $tasks = array();
    while ($row = $result->fetch_assoc()) {
        // Format dates for better readability
        if (isset($row['tanggal_upload'])) {
            $row['formatted_date'] = date('d M Y H:i', strtotime
                ($row['tanggal_upload']));
        }
        if (isset($row['created_at'])) {
            $row['formatted_created'] = date('d M Y H:i', strtotime
                ($row['created_at']));
        }

        $tasks[] = $row;
    }

    echo json_encode([
        'success' => true,
        'data' => $tasks
    ]);

    $stmt->close();
    $conn->close();

} catch (Exception $e) {
    echo json_encode([
        'success' => false,
        'message' => 'Error: ' . $e->getMessage()
    ]);
}
```

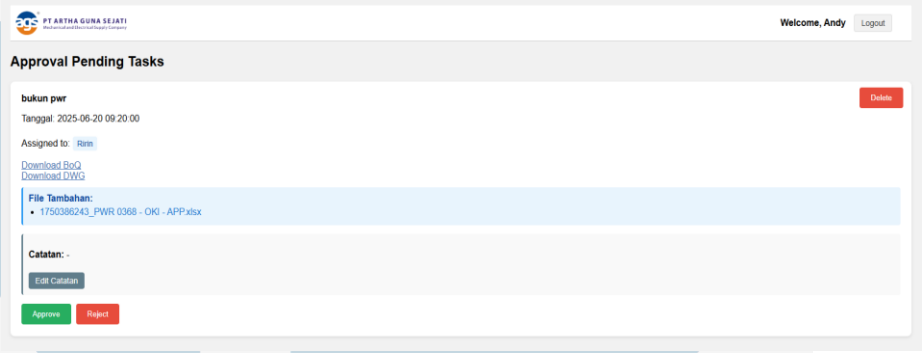
Gambar 3. 14 Kode PHP Halaman "Task History"

Gambar 3. 14 merupakan kode PHP halaman "*Task History"* yang merupakan potongan kode PHP pada gambar tersebut digunakan untuk menampilkan riwayat tugas (*task history*) yang telah dibuat oleh pengguna dengan role *super_admin*, khususnya pada halaman *manager* atau *head*. Pada bagian awal kode, dilakukan *query* SQL untuk mengambil data dari tabel *tasks* dan *admin*s, yang mencakup informasi seperti nama tugas, tanggal unggah, status, catatan, *file* BoQ, DWG, spek, penawaran, serta nama *admin* yang ditugaskan. Data ini diambil berdasarkan ID pembuat tugas (creator_id) dan diurutkan berdasarkan waktu pembuatan terbaru.

Setelah *query* disiapkan menggunakan prepared statement, parameter *user*Id dan limit dimasukkan ke dalam perintah SQL untuk memastikan data yang ditampilkan sesuai dengan identitas pengguna yang sedang *login* dan jumlah yang ingin ditampilkan. Kemudian hasil dari *query* diproses satu per satu, dan kolom tanggal seperti tanggal_*upload* dan created_at di*form*at agar lebih mudah dibaca (misalnya dalam *form*at "13 Jun 2025 15:30").

Semua data yang telah diproses ini dikembalikan dalam bentuk objek JSON yang berisi status keberhasilan (success: true) dan array data tugas. Nantinya, data JSON ini akan dibaca oleh kode *JavaScript* di sisi *frontend* untuk ditampilkan di halaman *task history manager*/*head*. Fitur ini sangat membantu pengguna dalam memantau seluruh tugas yang sudah pernah dibuat maupun yang masih berjalan, lengkap dengan informasi *admin* yang ditugaskan dan *file* yang diunggah.

Untuk mekanisme *approval*, dibuat sistem yang memungkinkan *user manager* melihat daftar tugas mereka sekaligus dapat mengambil keputusan persetujuan atau penolakan terhadap tugas tersebut. Apabila tombol approve ditekan, status tugas di *database* otomatis berubah menjadi "approved". Jika tugas ditolak, sistem akan meminta alasan penolakan yang kemudian disampaikan ke *admin* sebagai catatan untuk revisi dan perbaikan

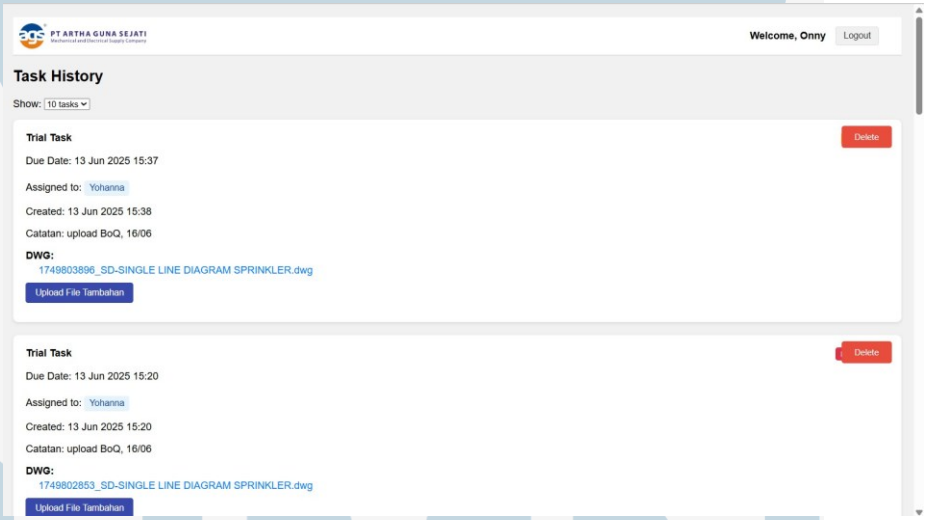


Gambar 3. 15 Halaman "Approval Pending Tasks"

Agar proses berjalan lancar, halaman ini juga dilengkapi dengan validasi *input*. Misalnya, pengguna wajib mengunggah minimal satu dokumen untuk dapat mengirimkan *form*. Selain itu, *noti*fikasi menggunakan *JavaScript* memberikan konfirmasi langsung kepada *admin* jika *file* berhasil diunggah atau jika ada kesalahan *input* yang harus diperbaiki. Pengguna juga dapat memantau riwayat status dari tugas-tugas yang pernah diunggah agar selalu mengetahui apakah dokumen sudah ditinjau *admin* atau masih dalam tahap proses.

Sebagai bagian dari fungsi pelengkap yang mendukung transparansi, halaman *manager* juga dilengkapi dengan fitur *task history*, yaitu tampilan riwayat tugas yang telah diunggah oleh *user head*. Fitur ini berfungsi sebagai alat monitoring agar pengguna dapat melihat daftar tugas-tugas yang sedang atau telah diproses oleh tim *admin*. Informasi yang ditampilkan mencakup nama tugas, tanggal pengunggahan, status terakhir (menunggu, disetujui, ditolak), catatan revisi dari *admin*, serta tanggal terakhir diperbarui. Dengan tampilan *visual* yang sederhana namun in*form*atif, *user* dapat dengan mudah mengetahui perkembangan.

Tampilan *task history* ini juga didukung dengan filter dan pencarian



Gambar 3. 16 Halaman "Task History"

berdasarkan tanggal, serta ikon status yang membedakan kategori tugas (approved, rejected, pending). Dengan adanya fitur ini, sistem menjadi lebih transparan dan memudahkan pelacakan dokumen oleh pihak kepala divisi.

Setelah semua fungsi utama selesai dikembangkan, dilakukan pengujian menyeluruh untuk memastikan kelancaran mulai dari pengisian *form*, proses penyimpanan *file*, hingga proses *approval*. Pada

tahap ini juga dilakukan penyesuaian tampilan antarmuka agar halaman menjadi lebih intuitif dan mudah digunakan. Beberapa aspek yang diperbaiki meliputi pengelompokan *file*, penggunaan ikon status yang jelas, serta penambahan filter berdasarkan tanggal supaya pencarian tugas lebih cepat dan efisien. Dengan adanya halaman *head* ini, proses dokumentasi dan komunikasi antara kepala divisi dan *admin* menjadi jauh lebih efektif dan terdokumentasi dengan baik. Sistem ini tidak hanya memudahkan pengelolaan dokumen, tetapi juga menciptakan alur kerja yang terstruktur, transparan, dan mudah dipantau oleh semua pihak terkait.

3.2.5 Pengujian & validasi *form upload* (*head*).

Tahap pengujian dan validasi pada *form upload* di halaman *head* merupakan bagian penting dalam proses pengembangan sistem *dashboard* internal, karena berfungsi sebagai penjaga kualitas untuk memastikan bahwa seluruh fungsi berjalan sesuai tujuan dan bahwa data yang masuk ke dalam sistem benar-benar valid dan lengkap. Halaman ini menjadi titik awal pengelolaan tugas dalam alur kerja antara kepala divisi dan tim *admin*, sehingga kestabilan dan ketepatan fungsinya sangat mempengaruhi kinerja sistem secara keseluruhan.

Proses pengujian diawali dengan memverifikasi setiap elemen *input* pada *form*, seperti *input* teks untuk nama tugas dan catatan tambahan, serta *input* unggahan untuk *file-file* yang terdiri dari dokumen BoQ, gambar teknis DWG, spesifikasi produk, dan *file* penawaran. Pengujian dilakukan untuk memastikan bahwa masing-masing elemen dapat menerima dan membaca data dengan benar sesuai dengan jenisnya. Misalnya, ketika pengguna *head* mengisi nama tugas dan mengunggah *file* DWG, sistem harus mampu mengenali *file* tersebut, menampilkanya secara lokal di daftar unggahan, dan menyimpanya dengan aman di server jika *form* dikirimkan.

Sistem secara otomatis akan mencegah *form* terkirim jika belum ada

dokumen yang dilampirkan, dan akan menampilkan peringatan kepada pengguna melalui *not*ifikasi *JavaScript*. Fitur ini sangat membantu dalam menjaga kualitas data dan mengurangi beban koreksi manual oleh *admin*.

Selain validasi elemen *form*, pengujian juga dilakukan pada proses *upload* itu sendiri. *File* yang dikirimkan oleh pengguna harus memenuhi sejumlah ketentuan, seperti batas ukuran maksimum *file* dan *form*at *file* yang didukung. Sistem dirancang untuk memberikan *feedback* secara real-time ketika pengguna mencoba mengunggah *file* dengan ekstensi yang tidak diizinkan, seperti .exe atau *file* lain di luar jenis dokumen teknis. Proses validasi juga mencakup uji coba terhadap *file* dengan ukuran besar untuk memastikan sistem mampu menangani unggahan tanpa terputus, serta memastikan bahwa data tersebut tetap tersimpan dengan benar di direktori server. Simulasi pengujian juga dilakukan dalam berbagai skenario, seperti ketika pengguna mencoba mengunggah banyak *file* sekaligus, mengunggah *file* yang sama lebih dari satu kali, atau mencoba mengirimkan *form* tanpa melengkapi informasi tertentu. Setiap skenario dirancang untuk mengukur bagaimana sistem merespons kemungkinan kesalahan pengguna dan apakah alur kerja tetap berjalan sesuai logika sistem. Dalam hal ini, sistem diuji untuk memastikan tidak ada data yang masuk ke *database* secara parsial atau tidak lengkap. Setelah *form* dikirimkan, sistem akan menyimpan data ke dalam dua tempat: *file* yang diunggah akan masuk ke dalam direktori *upload*s di server, sementara informasi seperti nama tugas, nama *file*, tanggal *upload*, status tugas, dan catatan dari pengguna akan dicatat ke dalam tabel *tasks* pada *database* MySQL. Status tugas akan secara otomatis

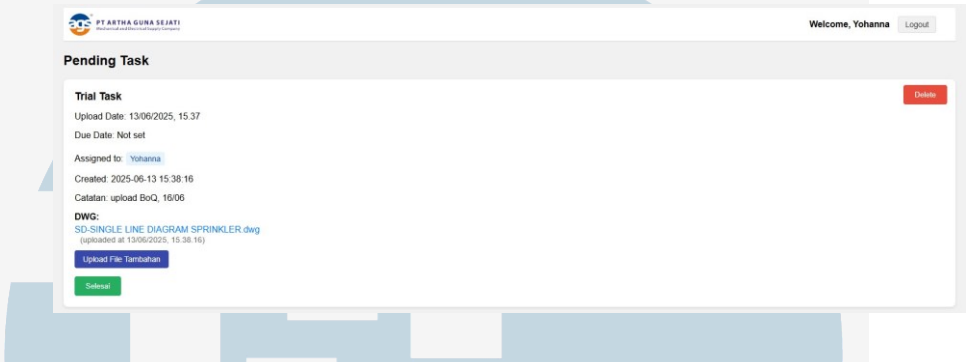
diatur ke "pending" sebagai status awal, menandakan bahwa tugas masih menunggu tindakan lanjutan dari *admin*. Proses penyimpanan ini juga diuji secara menyeluruh untuk memastikan tidak terjadi inkonsistensi antara data *file* dan metadata di *database*.

Hasil pengujian menunjukkan bahwa dengan implementasi validasi dan *feedback* real-time menggunakan *JavaScript*, pengguna lebih cepat mengetahui kesalahan *input*, memperbaikinya secara langsung, dan dapat menyelesaikan proses unggahan dengan lancar. Seluruh alur ini dirancang agar tidak hanya aman dari sisi teknis, tetapi juga memberikan pengalaman pengguna yang efisien dan minim hambatan. Dengan pengujian dan validasi yang ketat pada *form upload* ini, sistem dapat meminimalisir risiko terjadinya kesalahan *input*, kehilangan dokumen penting, dan menghindari miskomunikasi antara pengguna *head* dan *admin*. Tahap ini menjadi fondasi penting dalam memastikan kelancaran proses kerja internal perusahaan yang bergantung pada sistem digital.

3.2.6 Pengembangan Halaman *Admin*: Fitur revisi & melengkapi *task* (*admin mode*).

Fitur melengkapi *task* pada *mode admin* merupakan bagian dari sistem pengelolaan tugas yang dirancang untuk memastikan kelengkapan dokumen serta membantu mempercepat proses validasi dan finalisasi tugas. Dalam alur sistem ini, *admin* berperan sebagai pengelola dan pendukung operasional yang bertugas mengunggah dokumen tambahan apabila dibutuhkan, khususnya berdasarkan

permintaan dari pihak kepala divisi (*head*) atau *manager* yang sebelumnya telah memberikan instruksi melalui catatan revisi.



Gambar 3. 17 Halaman "Pending Task"

Ketika *admin* membuka detail sebuah tugas yang sebelumnya telah dikembalikan untuk revisi, sistem akan menampilkan seluruh informasi yang sudah ada, seperti nama tugas, waktu unggah, catatan, serta daftar dokumen yang telah diunggah sebelumnya, termasuk *file* BoQ, DWG, spesifikasi produk, dan penawaran harga. Jika dalam proses review ditemukan adanya kekurangan dokumen atau kebutuhan penambahan *file* tertentu yang telah dicatat oleh *head* atau *manager*, maka *admin* dapat menambahkan *file* tersebut langsung dari halaman detail tugas. Fungsi unggah ini tersedia khusus untuk melengkapi dokumen berdasarkan arahan yang sudah ada, bukan untuk membuat catatan revisi baru. Untuk memastikan bahwa *file* yang diunggah oleh *admin* tercatat secara resmi dalam sistem, maka *backend* akan melakukan proses pembaruan data ke dalam *database*. Berikut adalah bagian kode yang bertugas menyimpan informasi *file* ke kolom yang sesuai dalam tabel *tasks*.

```
// Prepare and execute the update query
$sql = "UPDATE tasks SET " . implode(", ", $updateFields) . " WHERE id = ?";
error_log("SQL: " . $sql);
```

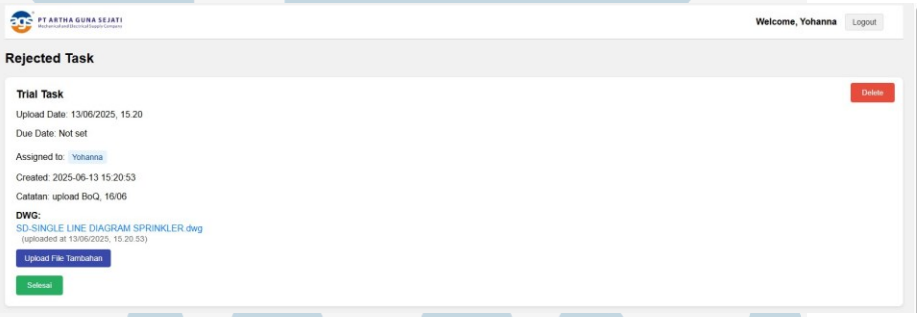
Gambar 3. 18 Kode PHP Update Task

Bagian ini merupakan inti dari proses update *file* pada sistem manajemen tugas. Setelah semua *file* berhasil diproses dan disimpan ke direktori server (*uploads*/), nama path *file* tersebut akan disiapkan untuk disimpan ke dalam *database* melalui *query* UPDATE. Kolom yang di-update bergantung pada jenis *file* yang berhasil diunggah, seperti boq, dwg, spek, penawaran, tambahan, maupun catatan tambahan.

Seluruh nama *file* dan catatan yang ingin diperbarui akan digabungkan sebagai parameter dalam *query*. *Query* UPDATE akan dijalankan untuk memperbarui *task* yang memiliki id sesuai dengan *task*Id. Jika berhasil, sistem akan mengembalikan respon JSON sukses yang menunjukkan bahwa *file* berhasil diperbarui. Bagian ini penting karena menunjukkan bagaimana sistem tidak hanya menyimpan *file* secara fisik di server, tetapi juga memperbarui informasi *file* secara dinamis ke *database* agar bisa ditampilkan kembali dalam halaman detail tugas oleh *user* terkait.

Selain itu juga terdapat komponen penting dalam proses ini adalah status Rejected *Task*, yaitu status yang muncul ketika sebuah tugas ditolak oleh kepala divisi atau *manager* setelah dilakukan peninjauan. Penolakan ini biasanya disebabkan oleh ketidaklengkapan dokumen, kesalahan *form*at, atau informasi yang tidak sesuai. Ketika sebuah tugas diberi status *rejected*, sistem secara otomatis akan memindahkan tugas tersebut ke halaman Rejected *Task*, baik di sisi *admin* maupun *head* atau *manager*. Pada saat proses penolakan berlangsung, kepala divisi atau *manager* dapat menambahkan catatan revisi yang berisi arahan atau instruksi

mengenai apa yang perlu diperbaiki, dilengkapi, atau diperbarui dalam dokumen tersebut. Catatan ini bersifat penting karena menjadi acuan utama bagi *admin* untuk mengetahui dokumen tambahan apa saja yang harus dilampirkan kembali ke dalam sistem. Setelah status tugas menjadi *rejected* dan catatan revisi disampaikan, *admin* dapat mengakses halaman detail tugas tersebut dan melihat seluruh *file* yang telah diunggah sebelumnya, serta membaca catatan revisi yang diberikan. Berdasarkan arahan tersebut, *admin* dapat mengunggah *file* tambahan yang dibutuhkan. Misalnya, jika kepala divisi meminta pelampiran ulang gambar teknis DWG atau menambahkan *file* spesifikasi produk yang sebelumnya belum lengkap, *admin* bisa mengunggah *file* tersebut langsung melalui halaman yang telah disediakan.



Gambar 3. 19 Halaman "Rejected Task"

Setiap *file* tambahan yang diunggah akan langsung tersimpan ke direktori server dan tercatat pada basis data agar riwayat kelengkapan tugas terdokumentasi dengan baik. *File-file* ini kemudian akan ditautkan secara otomatis ke tugas yang sedang dalam status revisi. Fitur ini memastikan bahwa semua dokumen pendukung dapat ditambahkan dengan aman dan tertelusur. Penting untuk dicatat bahwa dalam sistem ini, *admin* tidak memiliki akses untuk memberikan catatan revisi atau menolak tugas. Tindakan

tersebut sepenuhnya berada di bawah kewenangan kepala divisi atau *manager* yang memiliki tanggung jawab utama terhadap kualitas dan kelayakan dokumen proyek. Dengan demikian, fitur melengkapi dokumen berfungsi sebagai dukungan *admin*istratif dalam proses revisi yang telah diarahkan sebelumnya.

Pengujian terhadap fitur ini dilakukan untuk memastikan bahwa *admin* dapat mengunggah *file* tambahan secara tepat, *file* tersimpan dengan *form*at yang valid, dan keterkaitan *file* terhadap tugas dapat ditampilkan kembali dengan benar pada sistem. Selain itu, diuji juga bahwa sistem mencegah *admin* mengakses fungsi-fungsi revisi yang bersifat otoritatif, seperti mengubah status tugas atau memberikan catatan koreksi. Dengan adanya fitur ini, sistem memberikan fleksibilitas bagi *admin* untuk berkontribusi dalam penyempurnaan dokumen, menjaga kelengkapan berkas, dan mempercepat proses penyesuaian tanpa mengganggu struktur otorisasi yang telah ditetapkan. Alur kerja pun menjadi lebih terkoordinasi dengan jelas antara pihak pemberi tugas dan pihak pelaksana *admin*istratif.

3.2.7 Pengujian sistem menyeluruh (*head-admin flow*).

Pengujian sistem menyeluruh terhadap alur kerja antara pengguna berperan sebagai kepala divisi (*head*) dan *admin* merupakan salah satu tahapan krusial dalam pengembangan *dashboard* internal. Tahapan ini tidak hanya mengevaluasi masing-masing fungsi secara terpisah, tetapi juga menitikberatkan pada keterpaduan proses dan komunikasi antarperan yang terlibat dalam pengelolaan dokumen dan tugas. Fokus utama dari pengujian ini adalah memastikan bahwa sistem mampu menangani seluruh siklus kerja. Pengujian dimulai dari sisi *head*, yang memiliki tanggung jawab utama untuk menginisiasi alur kerja dengan mengunggah dokumen-

dokumen penting. Dokumen ini meliputi Bill of Quantity (BoQ), gambar teknis (DWG), spesifikasi produk, dan *file* penawaran harga, yang semuanya dilampirkan melalui *form upload* pada halaman *head*. Di sinilah validasi awal dari sistem diuji: setiap dokumen yang dikirimkan harus memenuhi persyaratan *form*at, ukuran *file*, dan kelengkapan data *input*. Sistem diharapkan mampu menolak pengiriman jika tidak ada *file* yang dilampirkan atau jika *form*at *file* tidak sesuai. Setelah *form* berhasil dikirim, sistem harus menyimpan *file* secara fisik ke direktori penyimpanan server, sementara metadata seperti nama tugas, nama *file*, waktu unggah, dan status awal "pending" akan dicatat secara otomatis ke dalam *database* PT_AGS pada tabel *tasks*.

Setelah proses unggahan oleh *head* selesai dan datanya tercatat dalam sistem, pengujian dilanjutkan dari sisi *admin*. *Admin* memiliki akses penuh terhadap daftar tugas yang masuk melalui halaman *backend* yang telah dirancang khusus untuk memfasilitasi proses verifikasi dan pengambilan keputusan. Dalam tahap ini, sistem diuji kemampuanya untuk menarik data dari *database* dan menampilkanya dalam bentuk tabel tugas yang mudah dibaca dan diurutkan berdasarkan waktu atau status. *Admin* kemudian melakukan pemeriksaan menyeluruh terhadap isi *file* yang diunggah, dan sistem harus memastikan bahwa semua *file* dapat dibuka tanpa error, serta mendeteksi jika terjadi ketidaksesuaian, seperti *file* kosong, *file* rusak, atau *form*at yang tidak relevan. Jika semua *file* sesuai dan lengkap, *admin* akan menggunakan tombol persetujuan yang secara otomatis mengubah status tugas menjadi "approved" di *database*. Pengujian memastikan bahwa status ini berubah secara real- time dan langsung terlihat oleh pengguna *head* di halaman mereka,

sehingga tidak terjadi keterlambatan atau kesalahan sinkronisasi. Namun, jika *admin* menemukan adanya kekurangan dalam dokumen, mereka dapat menekan tombol revisi dan mengisi catatan atau instruksi yang menjelaskan hal yang perlu diperbaiki. Dalam skenario ini, sistem diuji apakah catatan revisi dapat disimpan dengan benar dan ditampilkan jelas di halaman *head*, lengkap dengan waktu dan identitas *admin* yang memberikan *feedback*.

Lebih lanjut, pengujian juga melibatkan skenario di mana *admin* perlu menambahkan *file* pelengkap, seperti dokumen referensi tambahan atau versi baru dari dokumen yang sebelumnya tidak lengkap. Sistem diuji untuk memastikan *file* tambahan dapat dikaitkan dengan tugas yang relevan dan tidak menimpa *file* sebelumnya tanpa izin. Saat status tugas berubah menjadi “revisi”, sistem harus tetap menjaga data yang sudah tersimpan dan memfasilitasi *upload* ulang dari pihak *head*. Pada tahap ini, pengujian memastikan bahwa pengguna *head* dapat mengunggah versi revisi dari dokumen tanpa harus mengisi ulang semua elemen *form*, dan bahwa data baru berhasil menggantikan atau melengkapi dokumen lama sesuai instruksi *admin*.

Siklus ini bisa terjadi lebih dari satu kali, tergantung apakah dokumen hasil revisi masih membutuhkan koreksi lanjutan atau sudah sesuai. Karena itu, pengujian memastikan sistem mampu menangani proses revisi berulang tanpa konflik data, tanpa duplikasi yang membingungkan, dan tetap mencatat riwayat perubahan status secara kronologis. Setiap interaksi yang terjadi selama proses ini, mulai dari pengunggahan awal, revisi, *approval*, hingga finalisasi, diuji agar terekam dengan baik dalam log sistem dan dapat dilacak kembali jika dibutuhkan. Tim pengembang melakukan pengujian terhadap antarmuka untuk memastikan bahwa tombol-tombol seperti "Approve", "Revisi", "Unggah Ulang", dan "Lihat Catatan" berfungsi

dengan benar dan tampil secara konsisten di setiap perangkat. Layout halaman juga diuji agar navigasinya intuitif dan tidak menimbulkan kebingungan, terutama bagi pengguna non-teknis. Dalam semua kondisi tersebut, sistem harus mampu memberikan umpan balik yang jelas kepada pengguna, apakah dalam bentuk *not*ifikasi keberhasilan, peringatan kesalahan, atau instruksi lanjutan untuk memperbaiki *input* yang tidak valid.

Dengan selesainya pengujian menyeluruh terhadap alur kerja antara *head* dan *admin* ini, dapat disimpulkan bahwa sistem telah mampu menangani seluruh siklus manajemen tugas secara efisien, aman, dan fleksibel. Setiap proses yang terlibat, dari unggahan awal, validasi, revisi, hingga *approval* akhir, berjalan secara terpadu dan saling terhubung. Hasil dari pengujian ini menjadi dasar untuk memastikan bahwa sistem dapat diimplementasikan dalam lingkungan kerja perusahaan dengan tingkat keandalan dan konsistensi yang tinggi, serta mampu mendukung kolaborasi yang produktif antara kepala divisi dan tim *admin* dalam menangani berbagai jenis tugas dan dokumen proyek.

3.2.8 Bug fixing, UI refinement, *feedback*.

Setelah semua fitur utama selesai dikembangkan, tahap berikutnya adalah memperbaiki kesalahan (bug fixing), memperindah tampilan (UI refinement), dan mengolah masukan dari pengguna (*feedback*). Ketiga proses ini penting agar sistem bisa digunakan dengan nyaman, berjalan lancar, dan sesuai dengan kebutuhan pengguna sebenarnya. Selama sistem diuji oleh pengguna (baik *head* maupun *admin*), ditemukan beberapa masalah yang mengganggu jalanya fitur. Contoh bug yang ditemukan:

1. *File* tidak bisa diunggah, terutama jika ukuran *file* terlalu besar atau *form*atnya tidak sesuai.

2. Status tugas tidak langsung diperbarui, sehingga pengguna bingung apakah tugasnya sudah diproses atau belum.
3. Data tertimpa saat ada dua tugas dengan nama yang sama.
4. Validasi *form upload* tidak konsisten, misalnya *input* terlihat benar di halaman depan, tapi ditolak saat disimpan ke server.

Semua bug ini diperbaiki dengan cara mengecek ulang kode program di sisi *frontend* (HTML dan *JavaScript*) dan *backend* (PHP dan SQL). Setelah diperbaiki, dilakukan pengujian ulang untuk memastikan bug tidak muncul lagi dan fitur lainya tetap berjalan normal. Setelah sistem berjalan dengan benar, tampilan halaman juga disempurnakan agar lebih mudah digunakan dan terlihat profesional. Beberapa perubahan yang dilakukan:

1. Warna status ditambahkan, seperti hijau untuk tugas yang sudah disetujui, merah untuk revisi, dan kuning untuk tugas yang masih menunggu persetujuan. Ini memudahkan pengguna untuk mengenali status dokumen secara cepat.
2. Tata letak *form* diatur ulang, agar setiap bagian (nama tugas, unggah *file*, catatan) tertata rapi dan tidak membingungkan.
3. Tombol aksi diperjelas, seperti tombol *approve*, *revisi*, dan *lihat detail* dipisahkan agar tidak salah klik.
4. Teks dan ikon diperbesar, agar nyaman dilihat di berbagai ukuran layar (laptop, tablet, bahkan HP).
5. *Formulir* jadi lebih responsif, artinya tetap rapi dan terbaca meskipun dibuka di layar kecil.

Tujuan dari semua penyempurnaan ini adalah membuat pengguna lebih nyaman saat mengunggah dokumen, membaca status, atau melakukan tindakan tertentu. Setelah sistem diuji oleh pengguna asli (head dan admin), mereka diminta memberikan masukan tentang hal-hal yang masih kurang atau membingungkan. Beberapa *feedback* yang diterima antara lain:

1. *Admin* ingin fitur filter tugas, agar bisa mencari dokumen berdasarkan tanggal atau status.
2. *User head* ingin konfirmasi saat klik *upload*, supaya tidak salah kirim dokumen.
3. Beberapa pengguna minta ditambahkan log aktivitas, agar bisa melihat riwayat perubahan atau revisi yang pernah terjadi.
4. Desain tombol dan teks diminta agar lebih simpel dan langsung dipahami, terutama oleh pengguna yang tidak terbiasa menggunakan sistem digital.

Dari *feedback* tersebut, beberapa langsung diterapkan (seperti konfirmasi *upload* dan perbaikan tampilan), sementara lainya dijadikan catatan pengembangan lanjutan.

3.2.9 *Upload* ke hosting & integrasi *database*.

Setelah proses pengembangan sistem selesai dilakukan secara lokal (di komputer pengembang), langkah selanjutnya adalah mengunggah sistem ke server hosting dan mengintegrasikan dengan *database* online. Proses ini penting agar sistem bisa diakses oleh seluruh pengguna dari berbagai perangkat melalui jaringan internet. Langkah pertama adalah menyiapkan layanan web hosting yang mendukung PHP dan MySQL. Hosting yang digunakan dalam proyek ini adalah hosting standar dengan cPanel yang sudah menyediakan akses ke *File Manager*, PHPMy*Admin*, dan fitur domain/subdomain. Tahapan *upload* dilakukan sebagai berikut:

1. Mengunggah seluruh *file* sistem (HTML, CSS, *JavaScript*, dan PHP) ke dalam direktori public_html di server menggunakan *File Manager*.
2. Menyesuaikan path *file* dan koneksi agar URL dan direktori *file* yang digunakan sesuai dengan struktur folder di server hosting.

3. Menambahkan domain/subdomain jika diperlukan, sehingga pengguna bisa mengakses sistem melalui alamat URL tertentu, https://ptarthagunasejati.com/

Setelah semua *file* berhasil diunggah, dilakukan pengujian akses awal untuk memastikan bahwa tampilan halaman terbuka dengan benar di browser. Setelah halaman dapat diakses secara online, sistem *backend* yang sebelumnya berjalan dengan *database* lokal harus dihubungkan ke *database* MySQL yang berada di server hosting. Proses integrasi dilakukan dengan:

1. Membuat *database* baru melalui cPanel > MySQL *Database*, termasuk membuat *user* dan memberikan akses penuh terhadap *database* tersebut.
2. Mengimpor tabel dan data awal dari *database* lokal (biasanya berupa *file* .sql) ke PHPMy*Admin* di hosting, sehingga struktur tabel seperti *tasks* dan datanya tersedia di server.
3. Menyesuaikan *file* koneksi *database* (db_config.php) pada sistem, terutama bagian hostname, *username*, *password*, dan *database* name, agar mengarah ke *database* online.

```php
<?php
// Database Configuration
$host = "localhost";
$user = "###";
$pass = "###";
$db   = "###";

// Function to get database connection
function getDbConnection() {
    global $host, $user, $pass, $db;

    try {
        $dsn = "mysql:host=$host;dbname=$db";
        $pdo = new PDO($dsn, $user, $pass);
        $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        return $pdo;
    } catch (PDOException $e) {
        // Log error but don't expose details in production
        error_log("Database connection error: " . $e->getMessage());
        return false;
    }
}

// Function to get mysqli connection
function getMysqliConnection() {
    global $host, $user, $pass, $db;

    $conn = new mysqli($host, $user, $pass, $db);
    if ($conn->connect_error) {
        error_log("Database connection error: " . $conn->connect_error);
        return false;
    }
    return $conn;
}
?>
```

Gambar 3. 20 Database Configuration

Dalam proses pengembangan sistem berbasis web, *file* koneksi (koneksi.php) memiliki peran penting dalam menjembatani komunikasi antara aplikasi dan basis data. Pada *file* ini, terdapat dua fungsi utama, yaitu getDbConnection() dan getMysqliConnection(), yang masing-masing menggunakan metode PDO dan MySQLi untuk membangun koneksi ke *database*. Nilai-nilai tersebut perlu disesuaikan berdasarkan lingkungan tempat aplikasi dijalankan. Saat aplikasi dipindahkan ke server hosting, konfigurasi localhost harus diganti dengan hostname server online, begitu pula dengan *username*, *password*, dan nama *database*, agar aplikasi dapat mengakses *database* yang telah disediakan secara daring.

Fungsi getDbConnection() menggunakan pendekatan PDO dan dilengkapi dengan pengaturan *mode* error untuk memudahkan proses debugging, namun tetap menjaga keamanan dengan tidak menampilkan pesan kesalahan langsung kepada pengguna. Fungsi ini akan mengembalikan objek koneksi jika berhasil, atau false jika terjadi kesalahan.

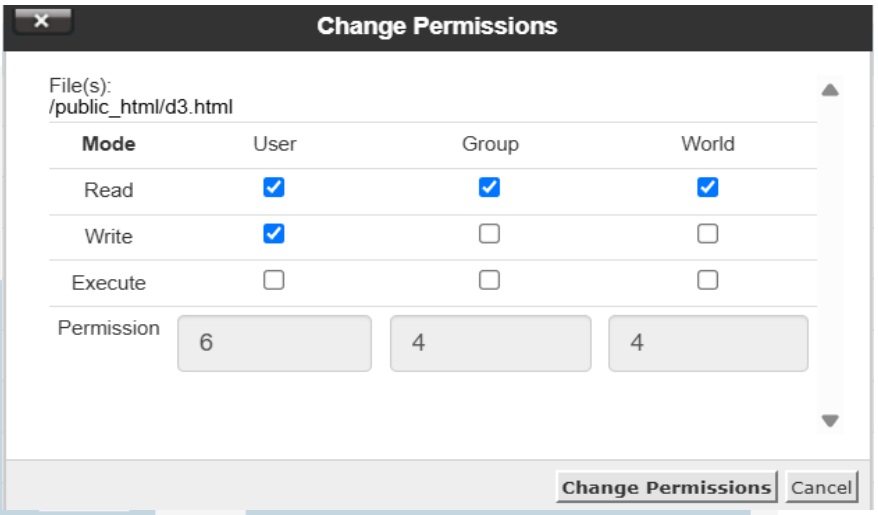
Sementara itu, fungsi getMysqliConnection() menggunakan pendekatan MySQLi. Jika koneksi gagal dilakukan, maka kesalahan akan dicatat ke dalam log melalui error_log, dan fungsi juga akan mengembalikan nilai false.

Dengan adanya dua pendekatan koneksi ini, sistem memiliki fleksibilitas dalam memilih metode koneksi yang paling sesuai. Penyesuaian terhadap *file* koneksi ini menjadi langkah krusial dalam proses deployment aplikasi, karena menentukan keberhasilan komunikasi antara sistem dan *database* secara daring.

4. Melakukan uji coba koneksi, dengan menjalankan halaman yang mengambil data dari *database* (seperti halaman *upload task* atau *task* list) untuk memastikan sistem berhasil menampilkan dan menyimpan data secara online.

Setelah koneksi berhasil, dilakukan beberapa penyesuaian tambahan seperti:

1. Menyesuaikan izin (permissions) folder agar *file upload* bisa disimpan di server.



Gambar 3. 21 Change Permissions

Dalam proses pengembangan dan deployment sistem web yang melibatkan fitur unggah dokumen, pengaturan izin akses (permissions) terhadap direktori server menjadi hal yang sangat penting. Tanpa pengaturan izin yang sesuai, sistem tidak akan memiliki hak untuk menyimpan *file* hasil unggahan ke dalam folder tujuan, yang dapat menyebabkan fitur *upload* gagal berfungsi. Gambar 3. 21 di atas menunjukkan tampilan pengaturan permissions pada panel hosting (cPanel), khususnya pada *file* /public_html/d3.html. Pada konfigurasi tersebut, pengaturan izin terbagi ke dalam tiga kategori pengguna, yaitu:

- *User* (pemilik *file*)
- Group (kelompok pengguna terkait)
- World (semua pengguna)

Setiap kategori dapat diberikan hak akses:

- Read (r): untuk membaca atau melihat isi *file*
- Write (w): untuk menulis atau memodifikasi *file*
- Execute (x): untuk mengeksekusi *file* sebagai program/script

Dalam contoh di atas, *file* diberikan hak akses sebagai berikut:

- *User*: Read & Write (izin penuh untuk membaca dan menulis)
- Group: Read
- World: Read

Pengaturan ini menghasilkan *mode* permission 644, yang merupakan standar umum untuk *file* HTML. Namun, untuk direktori penyimpanan *file upload*, seperti /public_html/*uploads*/, biasanya diperlukan permission 755 atau bahkan 775, agar skrip PHP memiliki izin untuk menulis ke dalam folder tersebut. Jika izin write tidak diberikan ke direktori tujuan, maka *file* hasil *upload* tidak akan bisa disimpan, meskipun proses *upload* di sisi client berhasil.

Dengan melakukan penyesuaian permission yang tepat, sistem dapat berjalan optimal, dan fitur *upload file* dapat berfungsi sebagaimana mestinya tanpa kendala hak akses. Langkah ini juga menjadi bagian dari proses konfigurasi akhir sebelum sistem dijalankan secara penuh di lingkungan hosting.

2. Menambahkan pengamanan dasar seperti validasi *input* dan batas ukuran *file*.
3. Melakukan simulasi penggunaan oleh *admin* dan *head* untuk memastikan semuanya berjalan baik dari sisi fungsional maupun jaringan.

Dengan diunggahnya sistem ke hosting dan berhasil terhubung ke *database* online, sistem kini siap digunakan oleh pengguna nyata.

3.2.10 Pemantauan sistem & perbaikan kecil.

Setelah sistem *dashboard* internal berhasil diunggah ke hosting dan diintegrasikan dengan *database* online, tahap selanjutnya adalah melakukan pemantauan berkala terhadap per*forma*n dan fungsionalitas

sistem selama digunakan oleh pengguna aktif, khususnya oleh tim *head* dan *admin*. Pemantauan sistem bertujuan untuk:

1. Memastikan bahwa seluruh fitur berjalan sesuai alur yang telah dirancang.
2. Menangkap secara dini kesalahan teknis atau bug yang mungkin tidak terdeteksi saat tahap uji coba lokal.
3. Menilai apakah tampilan antarmuka (UI) cukup nyaman dan responsif untuk digunakan di berbagai perangkat.
4. Mengumpulkan umpan balik dari pengguna untuk penyempurnaan lebih lanjut.

Pemantauan dilakukan melalui dua cara utama:

1. Observasi langsung terhadap penggunaan sistem oleh *user*, termasuk mencatat alur kerja saat *user* mengunggah *file*, menyetujui *task*, atau memberikan revisi.
2. Review teknis terhadap data dan error log, yaitu memeriksa *file* log di server, memantau *database* apakah data masuk dengan benar, serta mengecek apakah ada bagian halaman yang gagal dimuat atau tidak responsif.

Selama proses pemantauan, ditemukan beberapa hal yang perlu diperbaiki meskipun tidak bersifat kritis. Di antaranya:

1. Penempatan tombol *approval* dan revisi yang terlalu berdekatan, menyebabkan beberapa *user* salah klik. Solusinya, dilakukan pemisahan tombol dan diberi warna berbeda untuk membedakan fungsi secara *visual*.
2. Pesan *not*ifikasi yang terlalu singkat muncul, sehingga tidak sempat terbaca oleh pengguna. Perbaikanya adalah dengan memperpanjang durasi tampilnya *not*ifikasi dan menambahkan ikon atau warna yang mencolok.

3. *File* berukuran besar (terutama DWG) memerlukan waktu unggah yang cukup lama. Solusi yang diambil adalah menambahkan indikator progress bar agar *user* mengetahui status proses *upload* dan tidak mengira sistem error.
4. Responsivitas tampilan *form* pada layar perangkat mobile kurang optimal. Dilakukan penyesuaian CSS agar *form* dan tombol tetap mudah digunakan di layar kecil, termasuk penyesuaian font dan p*add*ing.
5. Kesalahan *input* dari *user*, seperti tidak memilih *file* atau tidak memberi nama tugas. Untuk mengatasi hal ini, ditambahkan validasi *JavaScript* yang memeriksa kelengkapan *input* sebelum proses dikirim ke server.

Perbaikan dilakukan langsung tanpa menghentikan sistem, karena sebagian besar bersifat front-end dan dapat diperbarui melalui update *file* di hosting. Berikut langkah-langkah perbaikan yang telah dilakukan:

1. Update layout halaman agar lebih rapi dan intuitif.
2. Revisi teks *not*ifikasi agar lebih jelas, misalnya mengganti "Sukses" menjadi "*File* berhasil diunggah!".
3. Penambahan pengecekan pada sisi client dan server untuk mencegah data kosong masuk ke *database*.
4. Penyesuaian warna tombol dan ikon status agar lebih mudah dibedakan antara pending, approved, dan revisi.
5. Menambahkan fitur filter berdasarkan tanggal untuk memudahkan pencarian tugas tertentu pada halaman *admin*.

Setelah dilakukan pemantauan dan perbaikan kecil tersebut, sistem menjadi lebih stabil dan lebih nyaman digunakan. Pengguna merasakan perbedaan dari sisi kemudahan dalam navigasi, kejelasan informasi, serta minimnya error saat penggunaan harian.

Dengan adanya proses pemantauan dan penyempurnaan ini, sistem tidak hanya berfungsi secara teknis, tetapi juga menunjukkan kesiapan untuk digunakan dalam lingkungan kerja nyata, mendukung komunikasi antar tim, serta memastikan bahwa proses dokumentasi proyek dapat berjalan dengan efisien dan terdokumentasi dengan baik.

3.3 Kendala yang Ditemukan

Selama proses kerja magang di PT Artha Guna Sejati, terdapat beberapa kendala dan kesulitan dalam mengembangkan *website company profile* serta sistem internal untuk pengelolaan penawaran proyek. Beberapa kendala utama yang dihadapi antara lain:

1. Dalam analisis kebutuhan sistem, terdapat perbedaan alur kerja antar divisi, sehingga menyulitkan perancangan sistem yang dapat memenuhi kebutuhan seluruh pengguna. Diperlukan diskusi dan kompromi antara divisi terkait agar sistem dapat berjalan secara optimal.
2. Pada tahap pengembangan, terdapat tantangan dalam menyambungkan *database* dengan *file* yang ingin diunggah ke dalam sistem. Proses integrasi ini memerlukan pengaturan khusus pada *backend* dan penyimpanan *file* agar dapat tersimpan dengan baik di *database* serta mudah diakses kembali. Beberapa penyesuaian dalam struktur *database* dan kode program dilakukan untuk memastikan *file* dapat terunggah dengan benar tanpa mengganggu kinerja sistem.
3. Kendala lainya muncul dalam pengujian sistem, terutama dalam memastikan bahwa setiap fitur dapat berjalan dengan baik di berbagai perangkat dan browser. Beberapa fungsi tidak berjalan optimal pada kondisi tertentu, sehingga diperlukan penyesuaian pada kode dan pengujian ulang untuk memastikan kompatibilitas sistem

3.4 Solusi atas Kendala yang Ditemukan

Untuk mengatasi kendala yang muncul selama proses kerja magang di PT Artha Guna Sejati, beberapa langkah solusi diterapkan sesuai dengan permasalahan yang dihadapi:

1. Diskusi lebih mendalam dilakukan dengan masing-masing divisi guna memahami alur kerja yang berbeda. Dengan mengumpulkan data dari berbagai pihak, sistem dirancang agar lebih fleksibel dan dapat digunakan oleh seluruh divisi tanpa mengubah prosedur kerja yang telah ada.
2. Kesulitan dalam menyambungkan *database* dengan *file* yang diunggah diatasi dengan menyesuaikan konfigurasi penyimpanan *file*. Salah satu pendekatan yang diterapkan adalah menggunakan sistem penyimpanan *file* terpisah yang tetap terhubung ke *database* melalui referensi *file* path. Selain itu, teknik validasi *file* sebelum diunggah diterapkan untuk memastikan integrasi berjalan dengan baik tanpa menimbulkan error pada sistem.
3. Pengujian lintas perangkat dan browser dilakukan guna mengidentifikasi perbedaan yang menyebabkan beberapa fitur tidak berjalan optimal. Setelah menemukan penyebab masalah, dilakukan penyesuaian kode dan debugging untuk memastikan semua fitur dapat berjalan dengan baik pada berbagai kondisi penggunaan.

Dengan menerapkan berbagai solusi tersebut, kendala yang dihadapi berhasil diatasi dan fungsionalitas sistem yang dikembangkan semakin meningkat. Selain itu, proses penyelesaian kendala ini juga memberikan pengalaman dan pemahaman yang lebih dalam mengenai pengembangan sistem berbasis web dan integrasi *database*.