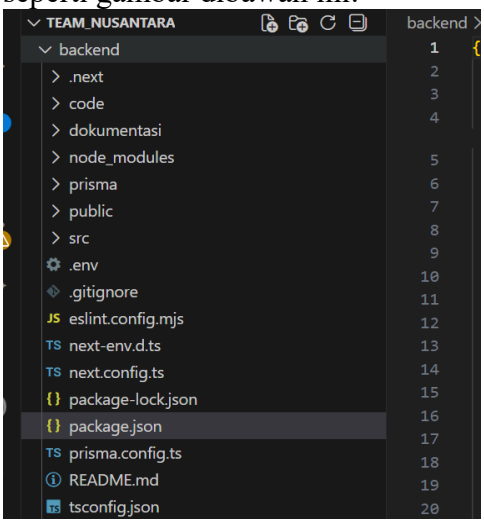
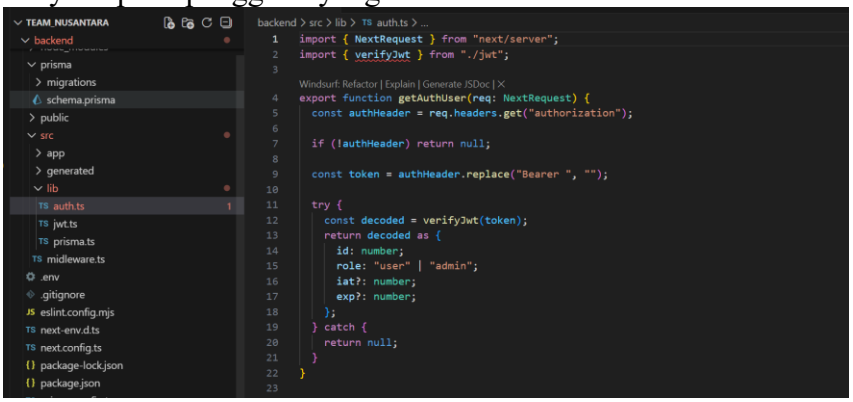
Nama : Muhammad Fabio Andre
Kelas : TI 23 A
Npm : 23313023

DOKUMENTASI PROJECT PWBL

1. Menginstall backend, setelah menginstall ubah port backend pada file package.json bagian scripts menjadi 3001 agar tidak sama dengan frontend nya dan membuat struktur folder seperti gambar dibawah ini:



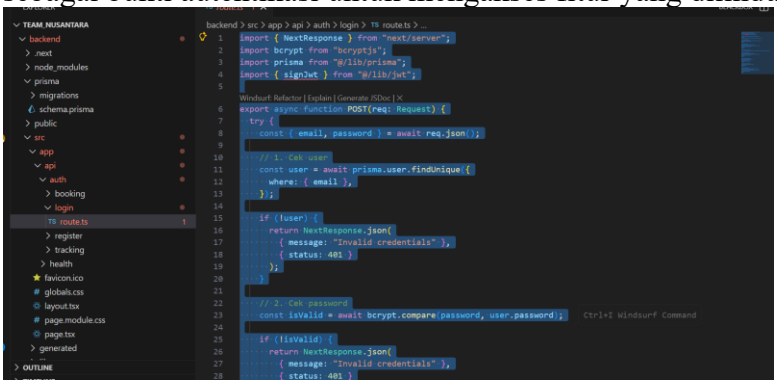
2. Pada file auth.ts di folder src/app kode ini digunakan untuk mengecek autentikasi pengguna melalui JWT pada setiap permintaan yang masuk. Fungsi getAuthUser mengambil token dari header Authorization, lalu memverifikasinya menggunakan verifyJwt. Jika token valid, data pengguna seperti id dan role akan dikembalikan. Namun, jika token tidak ada atau tidak valid, fungsi akan mengembalikan null, sehingga sistem dapat membatasi akses hanya kepada pengguna yang telah terautentikasi.



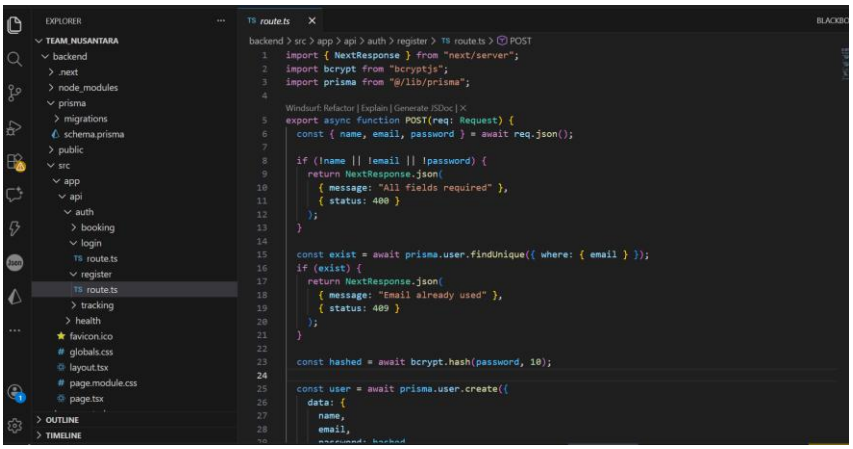
3. Pada file prisma.ts di folder src/app kode ini digunakan untuk menginisialisasi dan mengelola koneksi Prisma agar tidak membuat banyak instance saat aplikasi berjalan, terutama pada mode development. Dengan menyimpan objek PrismaClient di variabel global, aplikasi akan menggunakan satu koneksi yang sama meskipun terjadi reload. Hal

ini mencegah terjadinya error akibat terlalu banyak koneksi database dan membuat penggunaan Prisma lebih efisien serta stabil selama pengembangan:

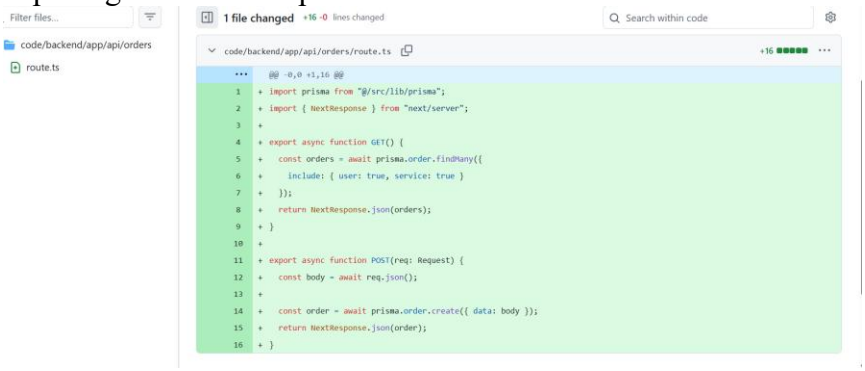
4. Pada file login/route.ts di folder src/app/api/auth kode ini merupakan API untuk proses login pengguna. Program menerima data email dan password dari request, lalu mengecek apakah pengguna dengan email tersebut ada di database menggunakan Prisma. Jika pengguna tidak ditemukan, sistem akan mengembalikan respons “Invalid credentials”. Selanjutnya, password yang dikirim akan dibandingkan dengan password yang tersimpan menggunakan bcrypt. Jika password benar, sistem akan membuat JWT yang berisi id dan role pengguna. Token ini kemudian dikirim kembali sebagai respons dan digunakan sebagai bukti autentikasi untuk mengakses fitur yang dilindungi:



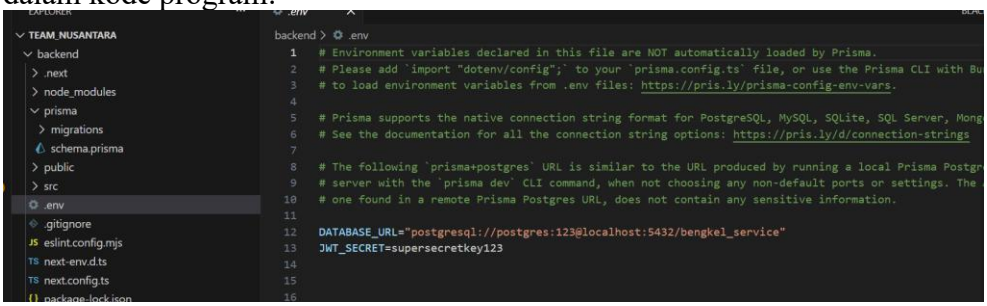
5. Pada file register/route.ts di folder src/app/api/auth kode ini digunakan sebagai API untuk proses pendaftaran (register) pengguna baru. Program menerima data name, email, dan password dari request, lalu memeriksa apakah semua data telah diisi. Selanjutnya, sistem mengecek apakah email sudah terdaftar di database. Jika belum, password akan dienkripsi menggunakan bcrypt sebelum disimpan. Data pengguna kemudian disimpan ke database melalui Prisma, dan sistem mengembalikan respons berupa id serta email pengguna yang berhasil didaftarkan:



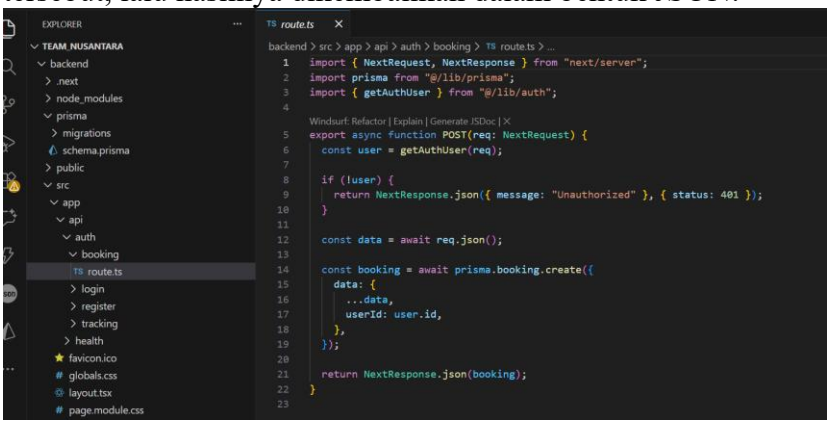
6. Kode ini digunakan sebagai API untuk mengelola data pesanan (*order*). Metode GET berfungsi mengambil seluruh data order dari database beserta relasi pengguna dan layanan, sedangkan metode POST digunakan untuk menambahkan data order baru berdasarkan data yang dikirim oleh client. Hasil proses tersebut dikembalikan dalam bentuk JSON agar dapat digunakan oleh aplikasi frontend:



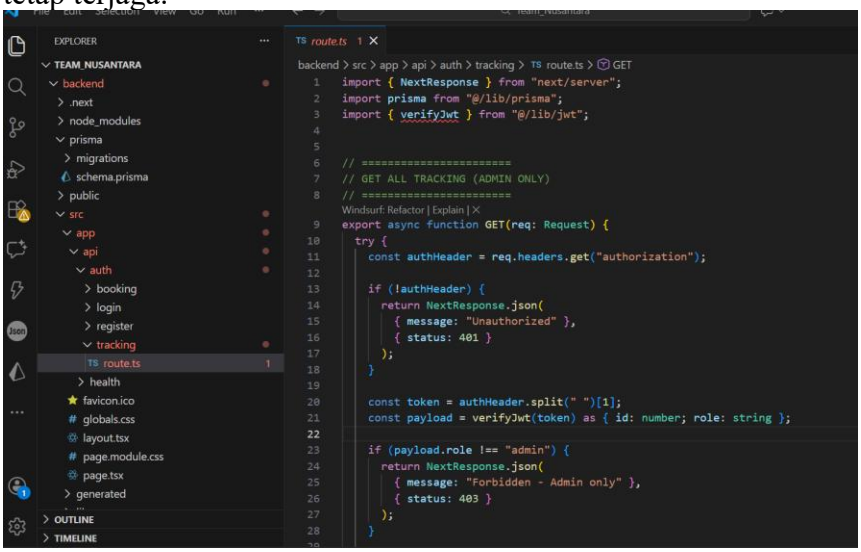
7. File ini digunakan untuk menyimpan variabel lingkungan (*environment variables*) yang dibutuhkan aplikasi. DATABASE_URL berisi konfigurasi koneksi ke database PostgreSQL, sedangkan JWT_SECRET digunakan sebagai kunci rahasia untuk membuat dan memverifikasi token JWT. Dengan memisahkan data penting ini ke dalam file .env, konfigurasi menjadi lebih aman dan fleksibel tanpa harus menuliskannya langsung di dalam kode program:



8. Kode ini digunakan sebagai API untuk membuat data booking yang hanya dapat diakses oleh pengguna yang telah terautentikasi. Fungsi getAuthUser digunakan untuk mengecek apakah request memiliki token yang valid. Jika pengguna belum login, sistem akan mengembalikan status *Unauthorized*. Jika sudah terautentikasi, data booking yang dikirim akan disimpan ke database melalui Prisma dengan menambahkan userId dari pengguna tersebut, lalu hasilnya dikembalikan dalam bentuk JSON:

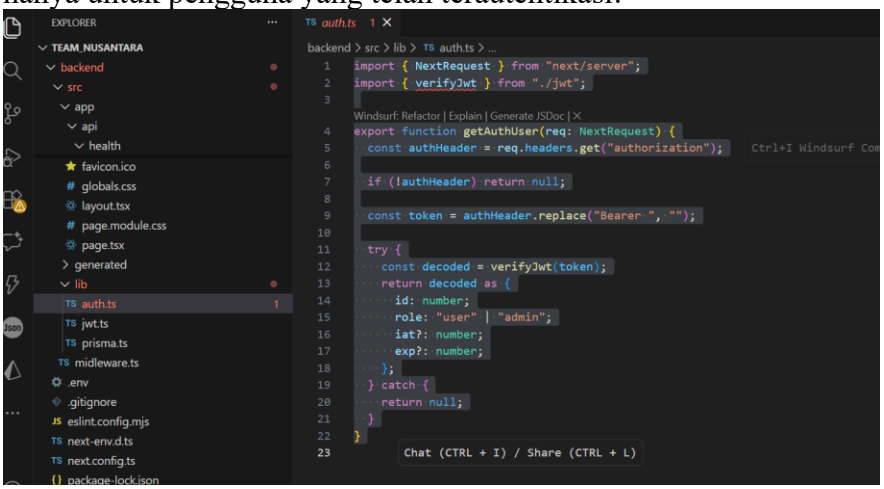


9. Kode ini merupakan API untuk mengelola data *tracking* yang hanya dapat diakses oleh admin. Pada metode GET, sistem memverifikasi token JWT dari header dan memastikan bahwa pengguna memiliki peran sebagai admin sebelum mengambil seluruh data tracking beserta informasi booking dan user. Pada metode POST, sistem juga memverifikasi bahwa permintaan berasal dari admin, kemudian membuat atau memperbarui data tracking berdasarkan bookingId dan progress yang dikirim. Dengan mekanisme ini, hanya admin yang dapat melihat dan mengubah status tracking, sehingga keamanan dan kontrol data tetap terjaga:



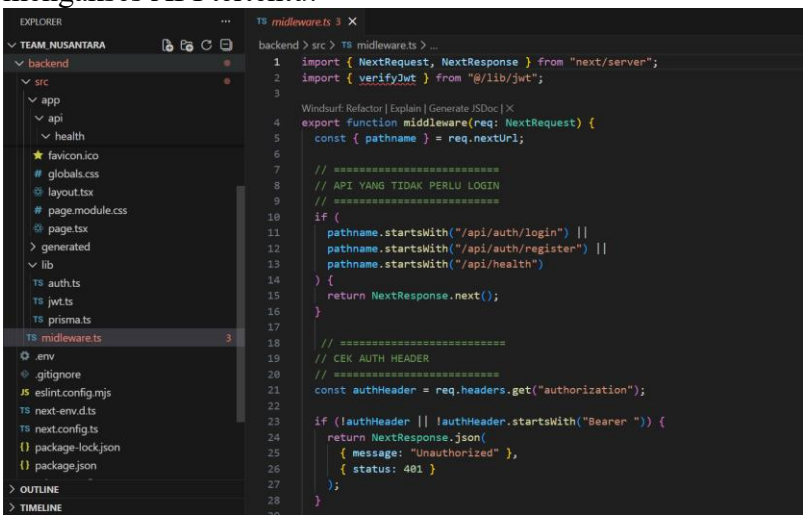
10. Kode ini berfungsi untuk membuat dan memverifikasi token JWT sebagai bagian dari sistem autentikasi. Variabel JWT_SECRET digunakan sebagai kunci rahasia untuk menjaga keamanan token. Fungsi signToken digunakan untuk menghasilkan token berisi data pengguna dengan masa berlaku satu hari, sedangkan verifyToken digunakan untuk memeriksa keaslian dan validitas token tersebut. Dengan mekanisme ini, sistem dapat memastikan bahwa setiap akses berasal dari pengguna yang sah dan terautentikasi:

11. Kode ini digunakan untuk mengambil dan memverifikasi data pengguna dari token JWT pada setiap request. Fungsi getAuthUser membaca header Authorization, mengambil token yang dikirim, lalu memverifikasinya menggunakan verifyJwt. Jika token valid, fungsi mengembalikan informasi pengguna seperti id dan role. Namun, jika token tidak ada atau tidak valid, fungsi akan mengembalikan null, sehingga sistem dapat membatasi akses hanya untuk pengguna yang telah terautentikasi:

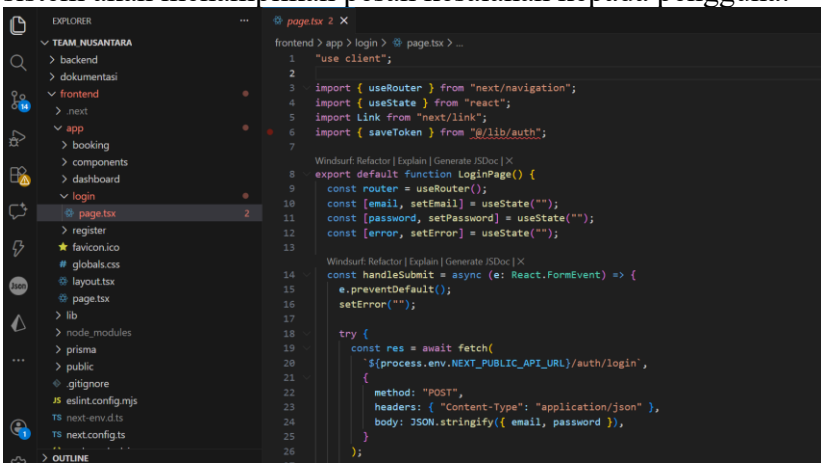


12. Kode middleware ini berfungsi sebagai pengaman API dengan memeriksa autentikasi menggunakan JWT. Middleware akan melewati beberapa endpoint publik seperti login, register, dan health tanpa pengecekan. Untuk endpoint lainnya, sistem memeriksa header

Authorization dan memverifikasi token. Jika token tidak ada atau tidak valid, akses akan ditolak. Selain itu, rute tertentu seperti /api/tracking dibatasi khusus untuk pengguna dengan peran admin. Dengan mekanisme ini, hanya pengguna yang berhak yang dapat mengakses API tertentu:

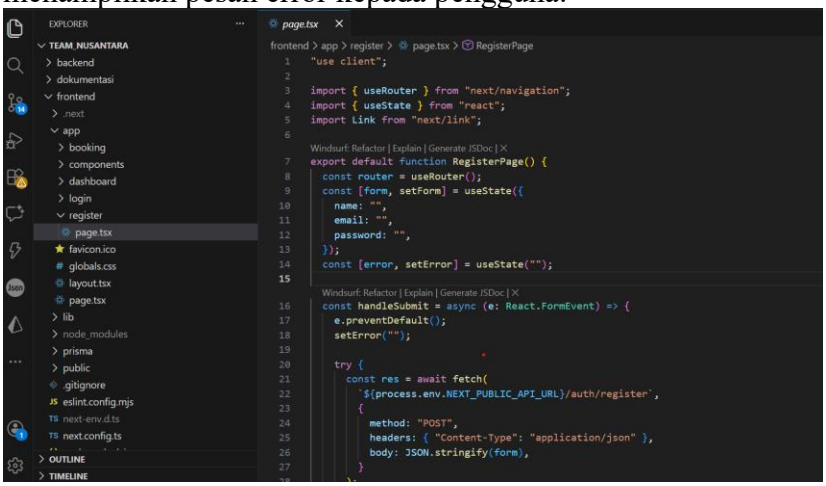


13. Kode ini merupakan komponen halaman login pada sisi frontend. Komponen ini sudah d hubunngkan ke backend dan menampilkan form untuk memasukkan email dan password, lalu mengirimkannya ke API saat tombol login ditekan. Jika proses login berhasil, token yang diterima akan disimpan menggunakan fungsi saveToken, kemudian pengguna diarahkan ke halaman dashboard. Apabila login gagal atau server tidak dapat dihubungi, sistem akan menampilkan pesan kesalahan kepada pengguna:



14. Kode ini merupakan komponen halaman pendaftaran pengguna pada sisi frontend. Codingan ini sudah di hubunngkan ke backend dan ini halaman menyediakan form untuk memasukkan nama, email, dan password, kemudian mengirimkan data tersebut ke API register saat form disubmit. Jika registrasi berhasil, pengguna akan diarahkan ke halaman

login. Namun, jika terjadi kesalahan atau server tidak dapat dihubungi, sistem akan menampilkan pesan error kepada pengguna:



15. Kode ini digunakan sebagai *layout* halaman dashboard yang hanya dapat diakses oleh pengguna yang sudah login. Saat komponen dimuat, sistem akan mengecek status login menggunakan fungsi isLoggedIn. Jika pengguna belum terautentikasi, maka akan diarahkan kembali ke halaman login. Jika sudah login, halaman dashboard akan ditampilkan lengkap dengan komponen Sidebar, Navbar, dan konten utama di dalamnya.