Dokumentasi Fitur: Authentikasi - Sign Up User Credential

> **Fokus Domain:** BACKEND
> **Konteks:** Trace Upstream ke Downstream secara Semantik

Alur Data Semantik (Scope: BACKEND)

```
@startuml
title Auth Register Flow (Upstream to Downstream)

actor Client

Client -> Server: HTTP POST /api/auth/register
Server -> Middleware: Routing & Middleware
Middleware -> Controller: Parsing Request & Delegation
Controller -> Service: Orchestrate Business Logic

Service -> UnitOfWork: Begin Transaction
UnitOfWork -> Repository: Data Access
Repository -> Mapper: Entity <-> Model Transformation
Mapper -> Model: Map to ORM Model
Model -> Database: Persist Data
Database --> Model: Persisted
Model --> Mapper
Mapper --> Repository
Repository --> UnitOfWork

UnitOfWork -> UnitOfWork: Commit Transaction
UnitOfWork --> Service

Service -> Mailer: Send OTP (Async)
note right of Mailer
Asynchronous process
Does not block response
end note

Service --> Controller: Registration Result
Controller --> Server: Build JSON Response
Server --> Client: HTTP Response JSON

@enduml
```

A. Laporan Implementasi Fitur Sign Up User Credential

Deskripsi Fungsional

Fitur ini menyediakan mekanisme pendaftaran akun pengguna baru dengan metode tradisional (email & password). Secara operasional, sistem menangani validasi keunikan identitas, pengamanan kredensial melalui algoritma hashing bcrypt, dan inisiasi siklus hidup akun dalam status "Pending". Untuk memitigasi risiko akun fiktif, sistem menerapkan mekanisme verifikasi dua langkah berbasis One-Time Password (OTP) yang dikirimkan ke alamat email terdaftar. Pengguna baru tidak dapat mengakses fitur utama aplikasi hingga proses verifikasi email berhasil diselesaikan.

Visualisasi

```
{
    "success": true,
    "code": 200,
    "message": "User registered successfully. Check console for OTP.",
    "data": {
        "id": "550e8400-e29b-41d4-a716-446655440000",
        "email": "user@example.com"
    }
}
```

Caption: Gambar 1: Struktur JSON Response sukses yang menandakan inisiasi data pengguna baru dan instruksi verifikasi.

B. Bedah Arsitektur & Komponen

Berikut adalah rincian 13 komponen yang menyusun fitur ini di sisi BACKEND, diurutkan dari upstream (penerima request) ke downstream (persistensi data dan notifikasi).

[internal/server/server.go](file:///d:/notetaker/notefiber-BE/internal/server/server.go)

Layer Terdeteksi: `HTTP Server & Route Registration`

Narasi Operasional: Komponen ini menginisialisasi instance server HTTP berbasis Fiber dan mendaftarkan seluruh middleware global (CORS, OpenTelemetry Tracing, Error Handler). Pada tahap bootstrap, ia menerima objek [Container](file:///d:/notetaker/notefiber-BE/internal/bootstrap/container.go#16-32) yang berisi semua controller yang sudah terinisialisasi, kemudian memanggil fungsi [registerRoutes](file:///d:/notetaker/notefiber-BE/internal/server/server.go#64-80) untuk menghubungkan setiap controller dengan path HTTP yang sesuai. Untuk alur Sign Up, [AuthController](file:///d:/notetaker/notefiber-BE/internal/controller/auth_controller.go#11-20) didaftarkan pada grup `/api`, sehingga endpoint `/api/auth/register` menjadi aktif dan siap menerima request dari klien.

```
func registerRoutes(app *fiber.App, c *bootstrap.Container) {
    api := app.Group("/api")

    c.AuthController.RegisterRoutes(api)
```

```
    c.UserController.RegisterRoutes(api)
    c.OAuthController.RegisterRoutes(api)
    // ... other controllers
}
```

Caption: Snippet 1: Registrasi route controller ke grup API utama.

[internal/bootstrap/container.go](file:///d:/notetaker/notefiber-BE/internal/bootstrap/container.go)

Layer Terdeteksi: `Dependency Injection Container`

Narasi Operasional: File ini mengorkestrasi konstruksi dan injeksi dependensi seluruh komponen aplikasi. Ia menginisialisasi infrastruktur inti ([RepositoryFactory](file:///d:/notetaker/notefiber-BE/internal/repository/unitofwork/repository_factory.go#5-8), [EmailService](file:///d:/notetaker/notefiber-BE/internal/pkg/mailer/email_service.go#11-15)) dan menyusun hierarki dependensi: Service dibangun dengan Repository Factory, kemudian Controller dibangun dengan Service. Untuk alur autentikasi, [AuthService](file:///d:/notetaker/notefiber-BE/internal/service/auth_service.go#26-35) diinisialisasi dengan `uowFactory` (untuk akses data) dan [emailService](file:///d:/notetaker/notefiber-BE/internal/pkg/mailer/email_service.go#16-21) (untuk pengiriman OTP), lalu diinjeksikan ke [AuthController](file:///d:/notetaker/notefiber-BE/internal/controller/auth_controller.go#11-20). Pendekatan ini memastikan decoupling antar layer dan memudahkan pengujian unit.

```
func NewContainer(db *gorm.DB, cfg *config.Config) *Container {
    // 1. Core Facades
    uowFactory := unitofwork.NewRepositoryFactory(db)
    emailService := mailer.NewEmailService(
        cfg.SMTP.Host, cfg.SMTP.Port, cfg.SMTP.Email, cfg.SMTP.Password,
cfg.SMTP.SenderName,
    )

    // 3. Services
    authService := service.NewAuthService(uowFactory, emailService)

    // 4. Controllers
    return &Container{
        AuthController: controller.NewAuthController(authService),
        // ...
    }
}
```

Caption: Snippet 2: Konstruksi hierarki dependensi untuk fitur autentikasi.

[internal/dto/auth_payment_dto.go](file:///d:/notetaker/notefiber-BE/internal/dto/auth_payment_dto.go)

Layer Terdeteksi: `Data Transfer Object (DTO)`

Narasi Operasional: File ini mendefinisikan kontrak data yang ketat untuk pertukaran informasi antara klien dan server. Struktur [RegisterRequest](file:///d:/notetaker/notefiber-BE/internal/dto/auth_payment_dto.go#10-15) menentukan field wajib (`full_name`, [email](file:///d:/notetaker/notefiber-BE/internal/pkg/mailer/email_service.go#16-21), `password`) beserta aturan validasi (menggunakan tag `validate`). Kontrak ini digunakan oleh Controller untuk mem-parsing body request dan memvalidasi format input sebelum data diteruskan ke layer Service. Struktur [RegisterResponse](file:///d:/notetaker/notefiber-BE/internal/dto/auth_payment_dto.go#21-25) mendefinisikan payload respons yang dikembalikan ke klien setelah proses registrasi berhasil.

```
type RegisterRequest struct {
    FullName string `json:"full_name" validate:"required,min=3"`
    Email    string `json:"email" validate:"required,email"`
    Password string `json:"password" validate:"required,min=8"`
}

type RegisterResponse struct {
    Id    uuid.UUID `json:"id"`
    Email string    `json:"email"`
}
```

Caption: Snippet 3: Definisi struktur data request dan response untuk registrasi.

[internal/controller/auth_controller.go](file:///d:/notetaker/notefiber-BE/internal/controller/auth_controller.go)

Layer Terdeteksi: `Interface / Controller Layer`

Narasi Operasional: Komponen ini menangani siklus Request-Response HTTP untuk seluruh endpoint autentikasi. Fungsi [RegisterRoutes](file:///d:/notetaker/notefiber-BE/internal/controller/auth_controller.go#12-13) mendaftarkan path `/auth/register` dengan handler [Register](file:///d:/notetaker/notefiber-BE/internal/service/auth_service.go#56-130). Saat request masuk, handler mem-parsing body JSON ke struktur [RegisterRequest](file:///d:/notetaker/notefiber-BE/internal/dto/auth_payment_dto.go#10-15), kemudian mendelegasikan eksekusi logika bisnis ke [AuthService](file:///d:/notetaker/notefiber-BE/internal/service/auth_service.go#26-35). Setelah proses selesai, ia membungkus hasil eksekusi ke dalam format respons JSON standar dengan field `success`, `code`, `message`, dan `data` yang konsisten.

```
func (c *authController) RegisterRoutes(r fiber.Router) {
    h := r.Group("/auth")
    h.Post("/register", c.Register)
    h.Post("/verify-email", c.VerifyEmail)
    // ...
}

func (c *authController) Register(ctx *fiber.Ctx) error {
    var req dto.RegisterRequest
    if err := ctx.BodyParser(&req); err != nil {
```

```
        return err
    }

    res, err := c.service.Register(ctx.Context(), &req)
    if err != nil {
        return ctx.Status(fiber.StatusBadRequest).JSON(fiber.Map{
            "success": false, "code": 400, "message": err.Error(),
        })
    }
    return ctx.JSON(fiber.Map{
        "success": true, "code": 200,
        "message": "User registered successfully. Check console for OTP.",
        "data": res,
    })
}
```

Caption: Snippet 4: Implementasi handler registrasi yang mengelola parsing, delegasi, dan formatting output.

[internal/service/auth_service.go](file:///d:/notetaker/notefiber-BE/internal/service/auth_service.go)

Layer Terdeteksi: `Business Logic / Service Layer`

Narasi Operasional: Komponen ini mengenkapsulasi inti logika pendaftaran. Ia mengoordinasikan serangkaian operasi kritis secara berurutan: (1) memvalidasi ketersediaan email melalui Unit of Work untuk mencegah duplikasi, (2) mengamankan password menggunakan algoritma bcrypt, (3) menyusun entitas User baru dengan status "Pending", (4) memulai transaksi database untuk menjamin atomisitas, (5) mempersistenkan data User dan Token Verifikasi secara bersamaan, (6) meng-commit transaksi, dan (7) memicu pengiriman email OTP secara asynchronous (goroutine) agar tidak memblokir respons ke klien. Ia bergantung pada [RepositoryFactory](file:///d:/notetaker/notefiber-BE/internal/repository/unitofwork/repository_factory.go#5-8) untuk akses data dan [IEmailService](file:///d:/notetaker/notefiber-BE/internal/pkg/mailer/email_service.go#11-15) untuk notifikasi.

```
func (s *authService) Register(ctx context.Context, req *dto.RegisterRequest)
(*dto.RegisterResponse, error) {
    uow := s.uowFactory.NewUnitOfWork(ctx)

    // 1. Check for existing user
    existing, _ := uow.UserRepository().FindOne(ctx, specification.ByEmail{Email:
req.Email})
    if existing != nil {
        return nil, errors.New("email already registered")
    }

    // 2. Hash password
    hash, err := bcrypt.GenerateFromPassword([]byte(req.Password),
bcrypt.DefaultCost)
    if err != nil { return nil, err }
```

```
    // 3. Create User Entity
    user := &entity.User{
        Id: uuid.New(), Email: req.Email, FullName: req.FullName,
        PasswordHash: &hashStr, Role: entity.UserRoleUser,
        Status: entity.UserStatusPending, EmailVerified: false,
    }

    // 4. Begin Transaction
    if err := uow.Begin(ctx); err != nil { return nil, err }
    defer uow.Rollback()

    // 5. Persist User & OTP Token
    if err := uow.UserRepository().Create(ctx, user); err != nil { return nil, err
}
    if err := uow.UserRepository().CreateEmailVerificationToken(ctx,
verificationToken); err != nil { return nil, err }

    // 6. Commit Transaction
    if err := uow.Commit(); err != nil { return nil, err }

    // 7. Async Email Delivery
    go func() { s.emailService.SendOTP(user.Email, otpCode) }()

    return &dto.RegisterResponse{Id: user.Id, Email: user.Email}, nil
}
```

Caption: Snippet 5: Orkestrasi logika bisnis registrasi dengan manajemen transaksi dan pengiriman email asynchronous.

[internal/repository/unitofwork/repository_factory.go](file:///d:/notetaker/notefiber-BE/internal/repository/unitofwork/repository_factory.go)

Layer Terdeteksi: `Factory Interface`

Narasi Operasional: File ini mendefinisikan kontrak untuk pembuatan instance Unit of Work. Interface [RepositoryFactory](file:///d:/notetaker/notefiber-BE/internal/repository/unitofwork/repository_factory.go#5-8) menyediakan satu metode [NewUnitOfWork](file:///d:/notetaker/notefiber-BE/internal/repository/unitofwork/repository_factory.go#6-7) yang menerima konteks dan mengembalikan instance [UnitOfWork](file:///d:/notetaker/notefiber-BE/internal/repository/unitofwork/unit_of_work.go#9-28) baru. Pendekatan ini memungkinkan Service layer untuk mendapatkan instance repository yang terisolasi per-request tanpa harus mengetahui detail implementasi koneksi database.

```
type RepositoryFactory interface {
    NewUnitOfWork(ctx context.Context) UnitOfWork
}
```

Caption: Snippet 6: Interface factory untuk pembuatan Unit of Work.

[internal/repository/unitofwork/repository_factory_impl.go](file:///d:/notetaker/notefiber-BE/internal/repository/unitofwork/repository_factory_impl.go)

Layer Terdeteksi: `Factory Implementation`

Narasi Operasional: Komponen ini mengimplementasikan [RepositoryFactory](file:///d:/notetaker/notefiber-BE/internal/repository/unitofwork/repository_factory.go#5-8) dengan menyimpan referensi ke koneksi database GORM. Saat [NewUnitOfWork](file:///d:/notetaker/notefiber-BE/internal/repository/unitofwork/repository_factory.go#6-7) dipanggil, ia membuat instance [UnitOfWork](file:///d:/notetaker/notefiber-BE/internal/repository/unitofwork/unit_of_work.go#9-28) baru yang siap digunakan untuk operasi database. Setiap Unit of Work bersifat short-lived (per-request) untuk menjamin isolasi transaksi antar request.

```
type RepositoryFactoryImpl struct {
    db *gorm.DB
}

func NewRepositoryFactory(db *gorm.DB) RepositoryFactory {
    return &RepositoryFactoryImpl{db: db}
}

func (f *RepositoryFactoryImpl) NewUnitOfWork(ctx context.Context) UnitOfWork {
    return NewUnitOfWork(f.db)
}
```

Caption: Snippet 7: Implementasi factory yang mengenkapsulasi koneksi database.

[internal/repository/unitofwork/unit_of_work.go](file:///d:/notetaker/notefiber-BE/internal/repository/unitofwork/unit_of_work.go)

Layer Terdeteksi: `Unit of Work Interface`

Narasi Operasional: File ini mendefinisikan kontrak untuk pola Unit of Work yang mengelola transaksi database dan menyediakan akses ke seluruh repository. Interface ini mendeklarasikan metode transaksional ([Begin](file:///d:/notetaker/notefiber-BE/internal/repository/unitofwork/unit_of_work_impl.go#31-38), [Commit](file:///d:/notetaker/notefiber-BE/internal/repository/unitofwork/unit_of_work_impl.go#39-47), [Rollback](file:///d:/notetaker/notefiber-BE/internal/repository/unitofwork/unit_of_work.go#12-13)) serta accessor untuk setiap repository domain ([UserRepository](file:///d:/notetaker/notefiber-BE/internal/repository/unitofwork/unit_of_work.go#14-15), [NotebookRepository](file:///d:/notetaker/notefiber-BE/internal/repository/unitofwork/unit_of_work_impl.go#63-66), dll). Dengan pola ini, Service layer dapat menjalankan beberapa operasi repository dalam satu transaksi atomik.

```
type UnitOfWork interface {
    Begin(ctx context.Context) error
    Commit() error
    Rollback() error
```

```
    UserRepository() contract.UserRepository
    NotebookRepository() contract.NotebookRepository
    NoteRepository() contract.NoteRepository
    // ... other repositories
}
```

Caption: Snippet 8: Interface Unit of Work untuk manajemen transaksi dan akses repository.

[internal/repository/unitofwork/unit_of_work_impl.go](file:///d:/notetaker/notefiber-BE/internal/repository/unitofwork/unit_of_work_impl.go)

Layer Terdeteksi: `Unit of Work Implementation`

Narasi Operasional: Komponen ini mengimplementasikan pola Unit of Work dengan memanfaatkan mekanisme transaksi GORM. Ia menyimpan referensi ke koneksi database (`db`) dan transaksi aktif (`tx`). Saat [Begin](file:///d:/notetaker/notefiber-BE/internal/repository/unitofwork/unit_of_work_impl.go#31-38) dipanggil, ia memulai transaksi baru. Accessor seperti [UserRepository()](file:///d:/notetaker/notefiber-BE/internal/repository/unitofwork/unit_of_work.go#14-15) mengembalikan instance repository yang terhubung ke transaksi aktif (jika ada) atau koneksi database biasa. Ini memastikan semua operasi repository dalam satu request berbagi transaksi yang sama.

```go
func (u *UnitOfWorkImpl) Begin(ctx context.Context) error {
    if u.tx != nil { return fmt.Errorf("transaction already started") }
    u.tx = u.db.WithContext(ctx).Begin()
    return u.tx.Error
}

func (u *UnitOfWorkImpl) Commit() error {
    if u.tx == nil { return fmt.Errorf("no transaction to commit") }
    err := u.tx.Commit().Error
    u.tx = nil
    return err
}

func (u *UnitOfWorkImpl) UserRepository() contract.UserRepository {
    return implementation.NewUserRepository(u.getDB())
}
```

Caption: Snippet 9: Implementasi manajemen transaksi dan instansiasi repository.

[internal/repository/contract/user_repository.go](file:///d:/notetaker/notefiber-BE/internal/repository/contract/user_repository.go)

Layer Terdeteksi: `Repository Interface / Contract`

Narasi Operasional: File ini mendefinisikan kontrak untuk seluruh operasi data terkait entitas User dan token-token terkait. Interface [UserRepository](file:///d:/notetaker/notefiber-

BE/internal/repository/unitofwork/unit_of_work.go#14-15) mendeklarasikan metode CRUD standar ([Create](file:///d:/notetaker/notefiber-BE/internal/repository/contract/user_repository.go#13-14), [Update](file:///d:/notetaker/notefiber-BE/internal/repository/contract/user_repository.go#14-15), [Delete](file:///d:/notetaker/notefiber-BE/internal/repository/implementation/user_repository_impl.go#55-58), [FindOne](file:///d:/notetaker/notefiber-BE/internal/repository/implementation/user_repository_impl.go#59-72), [FindAll](file:///d:/notetaker/notefiber-BE/internal/repository/implementation/user_repository_impl.go#73-83)) serta operasi spesifik seperti [CreateEmailVerificationToken](file:///d:/notetaker/notefiber-BE/internal/repository/contract/user_repository.go#27-28), [ActivateUser](file:///d:/notetaker/notefiber-BE/internal/repository/contract/user_repository.go#36-37), dan [CreateRefreshToken](file:///d:/notetaker/notefiber-BE/internal/repository/implementation/user_repository_impl.go#160-167). Kontrak ini memungkinkan Service layer berinteraksi dengan data tanpa mengetahui teknologi persistensi yang digunakan.

```
type UserRepository interface {
    Create(ctx context.Context, user *entity.User) error
    FindOne(ctx context.Context, specs ...specification.Specification)
(*entity.User, error)

    // Token Management
    CreateEmailVerificationToken(ctx context.Context, token
*entity.EmailVerificationToken) error
    FindEmailVerificationToken(ctx context.Context, specs
...specification.Specification) (*entity.EmailVerificationToken, error)
    DeleteEmailVerificationToken(ctx context.Context, id uuid.UUID) error

    // Business Specific
    ActivateUser(ctx context.Context, userId uuid.UUID) error
    // ...
}
```

Caption: Snippet 10: Kontrak repository untuk akses data User dan Token.

[internal/repository/specification/specification.go](file:///d:/notetaker/notefiber-BE/internal/repository/specification/specification.go)

Layer Terdeteksi: `Specification Pattern Interface`

Narasi Operasional: File ini mendefinisikan interface dasar untuk pola Specification yang digunakan untuk membangun query database secara deklaratif. Setiap specification mengimplementasikan metode [Apply](file:///d:/notetaker/notefiber-BE/internal/repository/specification/user_specifications.go#45-48) yang memodifikasi query GORM dengan kondisi tertentu. Pendekatan ini memisahkan logika query dari repository dan memungkinkan komposisi kondisi query yang fleksibel.

```
type Specification interface {
    Apply(db *gorm.DB) *gorm.DB
}
```

Caption: Snippet 11: Interface dasar untuk pola Specification.

[internal/repository/specification/user_specifications.go](file:///d:/notetaker/notefiber-BE/internal/repository/specification/user_specifications.go)

Layer Terdeteksi: `Specification Implementation`

Narasi Operasional: Komponen ini menyediakan implementasi specification khusus untuk domain User. [ByEmail](file:///d:/notetaker/notefiber-BE/internal/repository/specification/user_specifications.go#9-12) menambahkan filter berdasarkan alamat email, [UserOwnedBy](file:///d:/notetaker/notefiber-BE/internal/repository/specification/user_specifications.go#17-20) menambahkan filter berdasarkan ID user (digunakan untuk token), dan [ByToken](file:///d:/notetaker/notefiber-BE/internal/repository/specification/user_specifications.go#33-36) menambahkan filter berdasarkan nilai token. Specification ini digunakan oleh Service layer untuk membangun query pencarian yang spesifik tanpa menulis SQL secara langsung.

```
type ByEmail struct { Email string }
func (s ByEmail) Apply(db *gorm.DB) *gorm.DB {
    return db.Where("email = ?", s.Email)
}

type UserOwnedBy struct { UserID uuid.UUID }
func (s UserOwnedBy) Apply(db *gorm.DB) *gorm.DB {
    return db.Where("user_id = ?", s.UserID)
}

type ByToken struct { Token string }
func (s ByToken) Apply(db *gorm.DB) *gorm.DB {
    return db.Where("token = ?", s.Token)
}
```

Caption: Snippet 12: Implementasi specification untuk pencarian User dan Token.

[internal/repository/implementation/user_repository_impl.go](file:///d:/notetaker/notefiber-BE/internal/repository/implementation/user_repository_impl.go)

Layer Terdeteksi: `Repository Implementation`

Narasi Operasional: Komponen ini mengimplementasikan kontrak [UserRepository](file:///d:/notetaker/notefiber-BE/internal/repository/unitofwork/unit_of_work.go#14-15) dengan memanfaatkan GORM sebagai ORM. Ia menerima koneksi database dan menggunakan [UserMapper](file:///d:/notetaker/notefiber-BE/internal/mapper/user_mapper.go#8-9) untuk mentransformasi antara Entity (domain) dan Model (database). Metode [Create](file:///d:/notetaker/notefiber-BE/internal/repository/contract/user_repository.go#13-14) menerima entity User dari Service, mengkonversinya ke Model, mempersistenkan ke database, dan mengembalikan entity yang sudah diperkaya

dengan ID yang di-generate. Metode [CreateEmailVerificationToken](file:///d:/notetaker/notefiber-BE/internal/repository/contract/user_repository.go#27-28) mengikuti pola serupa untuk token verifikasi.

```
func (r *UserRepositoryImpl) Create(ctx context.Context, user *entity.User) error
{
    modelUser := r.mapper.ToModel(user)
    if err := r.db.WithContext(ctx).Create(modelUser).Error; err != nil {
        return err
    }
    *user = *r.mapper.ToEntity(modelUser)
    return nil
}

func (r *UserRepositoryImpl) CreateEmailVerificationToken(ctx context.Context,
token *entity.EmailVerificationToken) error {
    m := r.mapper.EmailVerificationTokenToModel(token)
    if err := r.db.WithContext(ctx).Create(m).Error; err != nil {
        return err
    }
    return nil
}

func (r *UserRepositoryImpl) FindOne(ctx context.Context, specs
...specification.Specification) (*entity.User, error) {
    var modelUser model.User
    query := r.applySpecifications(r.db.WithContext(ctx), specs...)
    if err := query.First(&modelUser).Error; err != nil {
        if errors.Is(err, gorm.ErrRecordNotFound) { return nil, nil }
        return nil, err
    }
    return r.mapper.ToEntity(&modelUser), nil
}
```

Caption: Snippet 13: Implementasi persistensi data dengan transformasi Entity-Model.

[internal/entity/user_entity.go](file:///d:/notetaker/notefiber-BE/internal/entity/user_entity.go)

Layer Terdeteksi: `Domain Entity`

Narasi Operasional: File ini mendefinisikan struktur data domain yang merepresentasikan konsep bisnis User dan token-token terkait. Entity [User](file:///d:/notetaker/notefiber-BE/internal/model/user_model.go#10-26) menyimpan atribut identitas (email, nama), kredensial (password hash), status akun, dan metadata audit. Entity [EmailVerificationToken](file:///d:/notetaker/notefiber-BE/internal/model/user_model.go#57-64) merepresentasikan token OTP dengan masa berlaku. Entity ini digunakan eksklusif di layer Service dan Repository, terpisah dari representasi database (Model) dan representasi API (DTO).

```
type UserRole string
type UserStatus string
```

```
const (
    UserRoleUser  UserRole = "user"
    UserRoleAdmin UserRole = "admin"

    UserStatusPending UserStatus = "pending"
    UserStatusActive  UserStatus = "active"
    UserStatusBlocked UserStatus = "blocked"
)

type User struct {
    Id            uuid.UUID
    Email         string
    PasswordHash  *string
    FullName      string
    Role          UserRole
    Status        UserStatus
    EmailVerified bool
    // ...
}

type EmailVerificationToken struct {
    Id        uuid.UUID
    UserId    uuid.UUID
    Token     string
    ExpiresAt time.Time
    CreatedAt time.Time
}
```

Caption: Snippet 14: Definisi Entity domain untuk User dan Token Verifikasi.

[internal/model/user_model.go](file:///d:/notetaker/notefiber-BE/internal/model/user_model.go)

Layer Terdeteksi: `Database Model (ORM)`

Narasi Operasional: Komponen ini mendefinisikan struktur data yang dipetakan langsung ke tabel database menggunakan tag GORM. Model [User](file:///d:/notetaker/notefiber-BE/internal/model/user_model.go#10-26) dipetakan ke tabel `users` dengan konfigurasi kolom (tipe data, constraint, default value) yang eksplisit. Model [EmailVerificationToken](file:///d:/notetaker/notefiber-BE/internal/model/user_model.go#57-64) dipetakan ke tabel `email_verification_tokens`. Pemisahan Model dari Entity memungkinkan evolusi skema database tanpa mempengaruhi logika bisnis.

```
type User struct {
    Id            uuid.UUID
`gorm:"type:uuid;primaryKey;default:gen_random_uuid()"`
    Email         string         `gorm:"type:varchar(255);uniqueIndex;not null"`
    PasswordHash  *string        `gorm:"type:varchar(255)"`
    FullName      string         `gorm:"type:varchar(255);not null"`
    Role          string         `gorm:"type:varchar(50);not null;default:'user'"`
    Status        string         `gorm:"type:varchar(50);not
```

```
null;default:'pending'"`
    EmailVerified bool           `gorm:"default:false"`
    // ...
}

func (User) TableName() string { return "users" }

type EmailVerificationToken struct {
    Id        uuid.UUID `gorm:"type:uuid;primaryKey;default:gen_random_uuid()"`
    UserId    uuid.UUID `gorm:"type:uuid;not null;index"`
    Token     string    `gorm:"type:varchar(255);not null;index"`
    ExpiresAt time.Time `gorm:"not null"`
    CreatedAt time.Time `gorm:"autoCreateTime"`
}

func (EmailVerificationToken) TableName() string { return
"email_verification_tokens" }
```

Caption: Snippet 15: Model ORM dengan mapping ke tabel database.

[internal/mapper/user_mapper.go](file:///d:/notetaker/notefiber-BE/internal/mapper/user_mapper.go)

Layer Terdeteksi: `Data Mapper`

Narasi Operasional: Komponen ini menyediakan fungsi konversi dua arah antara Entity (domain) dan Model (database). [ToModel](file:///d:/notetaker/notefiber-BE/internal/mapper/user_mapper.go#35-55) mentransformasi Entity User menjadi Model User sebelum persistensi, termasuk konversi tipe enum ke string. [ToEntity](file:///d:/notetaker/notefiber-BE/internal/mapper/user_mapper.go#14-34) melakukan transformasi sebaliknya setelah data diambil dari database. Mapper juga menyediakan fungsi serupa untuk token ([EmailVerificationTokenToModel](file:///d:/notetaker/notefiber-BE/internal/mapper/user_mapper.go#143-155), [EmailVerificationTokenToEntity](file:///d:/notetaker/notefiber-BE/internal/mapper/user_mapper.go#130-142)). Pemisahan ini menjaga kemurnian layer domain dari concern persistensi.

```
func (m *UserMapper) ToEntity(u *model.User) *entity.User {
    if u == nil { return nil }
    return &entity.User{
        Id:            u.Id,
        Email:         u.Email,
        PasswordHash:  u.PasswordHash,
        FullName:      u.FullName,
        Role:          entity.UserRole(u.Role),
        Status:        entity.UserStatus(u.Status),
        EmailVerified: u.EmailVerified,
        // ...
    }
}

func (m *UserMapper) ToModel(u *entity.User) *model.User {
```

```
    if u == nil { return nil }
    return &model.User{
        Id:            u.Id,
        Email:         u.Email,
        PasswordHash:  u.PasswordHash,
        Role:          string(u.Role),
        Status:        string(u.Status),
        // ...
    }
}

func (m *UserMapper) EmailVerificationTokenToModel(t
*entity.EmailVerificationToken) *model.EmailVerificationToken {
    // ... transformation logic
}
```

Caption: Snippet 16: Transformasi bidirectional antara Entity dan Model.

[internal/pkg/mailer/email_service.go](file:///d:/notetaker/notefiber-BE/internal/pkg/mailer/email_service.go)

Layer Terdeteksi: `Infrastructure / External Service Adapter`

Narasi Operasional: Komponen ini mengorkestrasi pengiriman email transaksional menggunakan protokol SMTP. Interface [IEmailService](file:///d:/notetaker/notefiber-BE/internal/pkg/mailer/email_service.go#11-15) mendefinisikan kontrak untuk pengiriman OTP dan Reset Token. Implementasi [emailService](file:///d:/notetaker/notefiber-BE/internal/pkg/mailer/email_service.go#16-21) mengenkapsulasi konfigurasi SMTP dialer dan menyediakan template HTML untuk setiap jenis email. Metode [SendOTP](file:///d:/notetaker/notefiber-BE/internal/pkg/mailer/email_service.go#35-61) dipanggil secara asynchronous oleh [AuthService](file:///d:/notetaker/notefiber-BE/internal/service/auth_service.go#26-35) setelah transaksi database berhasil di-commit, sehingga kegagalan pengiriman email tidak mempengaruhi keberhasilan registrasi.

```
type IEmailService interface {
    SendOTP(toEmail, otp string) error
    SendResetToken(toEmail, token string) error
}

func (s *emailService) SendOTP(toEmail, otp string) error {
    m := gomail.NewMessage()
    m.SetHeader("From", s.senderEmail)
    m.SetHeader("To", toEmail)
    m.SetHeader("Subject", "Your Verification Code")

    body := fmt.Sprintf(`
        <div style="font-family: Arial, sans-serif; padding: 20px;">
            <h2>Welcome to NoteFiber!</h2>
            <p>Your verification code is:</p>
            <h1 style="color: #4CAF50; letter-spacing: 5px;">%s</h1>
```

```
            <p>This code will expire in 15 minutes.</p>
        </div>
    `, otp)

    m.SetBody("text/html", body)
    return s.dialer.DialAndSend(m)
}
```

Caption: Snippet 17: Implementasi pengiriman email OTP dengan template HTML.

C. Ringkasan Layer Arsitektur

No	Layer	File	Tanggung Jawab
1	HTTP Server	[server/server.go](file:///d:/notetaker/notefiber-BE/internal/server/server.go)	Inisialisasi Fiber, middleware, route registration
2	DI Container	[bootstrap/container.go](file:///d:/notetaker/notefiber-BE/internal/bootstrap/container.go)	Dependency wiring & injection
3	DTO	[dto/auth_payment_dto.go](file:///d:/notetaker/notefiber-BE/internal/dto/auth_payment_dto.go)	Kontrak data request/response
4	Controller	[controller/auth_controller.go](file:///d:/notetaker/notefiber-BE/internal/controller/auth_controller.go)	HTTP handler, parsing, response formatting
5	Service	[service/auth_service.go](file:///d:/notetaker/notefiber-BE/internal/service/auth_service.go)	Orkestrasi logika bisnis
6	Factory Interface	[unitofwork/repository_factory.go] (file:///d:/notetaker/notefiber-BE/internal/repository/unitofwork/repository_factory.go)	Kontrak pembuatan Unit of Work
7	Factory Impl	[unitofwork/repository_factory_impl.go] (file:///d:/notetaker/notefiber-BE/internal/repository/unitofwork/repository_factory_impl.go)	Implementasi factory
8	UoW Interface	[unitofwork/unit_of_work.go](file:///d:/notetaker/notefiber-BE/internal/repository/unitofwork/unit_of_work.go)	Kontrak transaksi & akses repository
9	UoW Impl	[unitofwork/unit_of_work_impl.go] (file:///d:/notetaker/notefiber-BE/internal/repository/unitofwork/unit_of_work_impl.go)	Manajemen transaksi GORM

No	Layer	File	Tanggung Jawab
10	Repository Contract	[contract/user_repository.go](file:///d:/notetaker/notefiber-BE/internal/repository/contract/user_repository.go)	Interface akses data User
11	Specification Base	[specification/specification.go](file:///d:/notetaker/notefiber-BE/internal/repository/specification/specification.go)	Interface pola specification
12	User Specifications	[specification/user_specifications.go] (file:///d:/notetaker/notefiber-BE/internal/repository/specification/user_specifications.go)	Query predicates untuk User
13	Repository Impl	[implementation/user_repository_impl.go] (file:///d:/notetaker/notefiber-BE/internal/repository/implementation/user_repository_impl.go)	Persistensi data dengan GORM
14	Entity	[entity/user_entity.go](file:///d:/notetaker/notefiber-BE/internal/entity/user_entity.go)	Objek domain
15	Model	[model/user_model.go](file:///d:/notetaker/notefiber-BE/internal/model/user_model.go)	Representasi tabel database
16	Mapper	[mapper/user_mapper.go](file:///d:/notetaker/notefiber-BE/internal/mapper/user_mapper.go)	Transformasi Entity <-> Model
17	Mailer	[pkg/mailer/email_service.go](file:///d:/notetaker/notefiber-BE/internal/pkg/mailer/email_service.go)	Pengiriman email SMTP

Dokumen ini di-generate dalam mode READ-ONLY tanpa modifikasi terhadap kode sumber.