CS 319 Object-Oriented Software Engineering

Project Deliverable 6 (D6) - S2T7 - agora

BTO Core Developer Manual

Team Members:

- Bertan Uran - 22102541
- Egehan Yıldız - 22203014
- Ekin Köylü - 22103867
- Emre Yazıcıoğlu - 22201668
- İlke Latifoğlu - 22203818
- Merve Güleç - 22103231

CONTENTS

Project Overview

BTO Core is a comprehensive tour management system designed for Bilkent Tanıtım Ofisi. It handles tour scheduling, guide management, feedback collection, and real-time location tracking for campus tours.

Key Features

- User authentication with role-based access control
- Tour scheduling and management
- Guide assignment and tracking
- Real-time location monitoring
- Feedback collection and management
- Email notifications
- Profile management

Getting Started

Prerequisites

- Node.js (v14 or higher)
- PostgreSQL
- MapBox API Key

Environment Setup

1. Clone the repository

   ```
   git clone https://github.com/CS319-24-FA/S2-T7-agora
   ```

2. Install dependencies

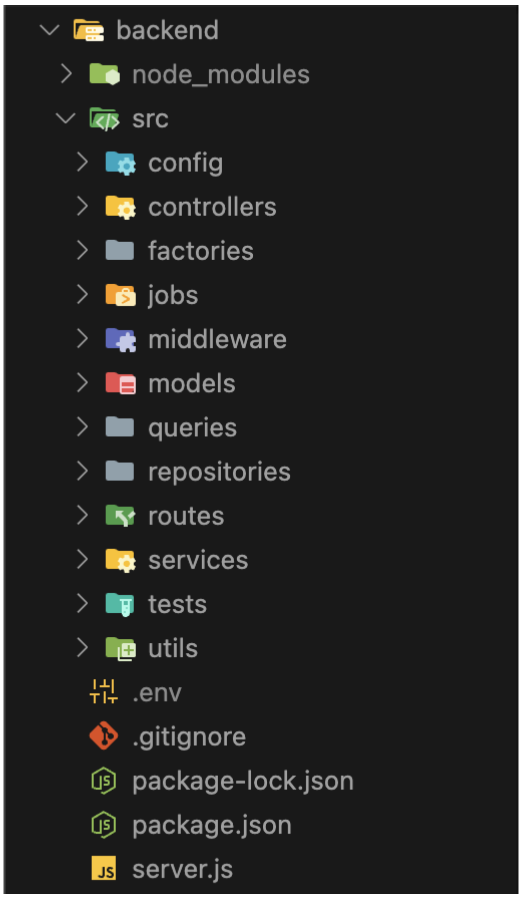
   ```
   cd backend
   npm install
   cd ../frontend
   npm install
   ```

3. Configure environment variables: Create a .env file in the backend directory with the following variables:

   ```
   PORT=3001
   DATABASE_URL=postgresql://[user]:[password]@localhost:5432/bto_core
   JWT_SECRET=[your-jwt-secret]
   FRONTEND_URL=http://localhost:3000
   MAPBOX_TOKEN=[mapbox-token]
   ```

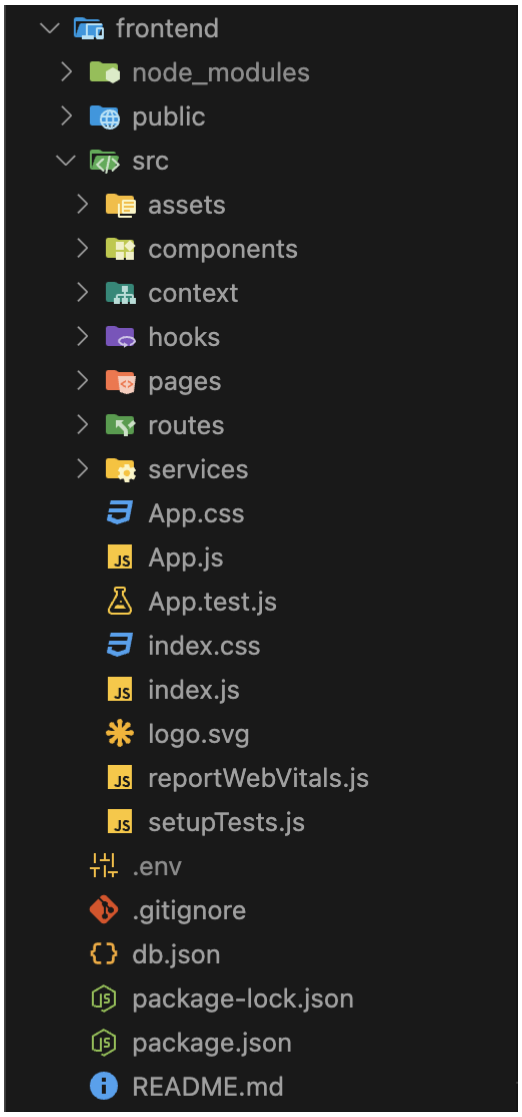
1. Project Structure

1.1 Backend Structure



```
backend
  node_modules
  src
    config
    controllers
    factories
    jobs
    middleware
    models
    queries
    repositories
    routes
    services
    tests
    utils
  .env
  .gitignore
  package-lock.json
  package.json
  server.js
```

1.2 Frontend Structure



```
frontend
  node_modules
  public
  src
    assets
    components
    context
    hooks
    pages
    routes
    services
    App.css
    App.js
    App.test.js
    index.css
    index.js
    logo.svg
    reportWebVitals.js
    setupTests.js
  .env
  .gitignore
  db.json
  package-lock.json
  package.json
  README.md
```

2. Backend Structure

2.1 `config/`

Contains configuration files including database setup (`database.js`).

2.2 `controllers/`

Houses the application's business logic handlers:

- `authController.js` - Authentication operations
- `tourController.js` - Tour management
- `feedbackController.js` - Feedback handling

2.2.1 `authController.js/`

`register(req, res)`

- Handles user registration
- Validates Bilkent email domain
- Creates user with hashed password
- Sends registration email
- Supports different user roles (advisor, candidate guide)

`login(req, res)`

- Validates credentials
- Generates OTP
- Sends OTP email
- Returns temporary token

`verifyOtp(req, res)`

- Verifies OTP token
- Issues permanent JWT token

2.2.2 `tourController.js/`

`addTour(req, res)`

- Creates new tour

- Validates school existence
- Handles time preferences
- Sends confirmation email

`assignGuideToTour(req, res)`

- Assigns guide to tour
- Checks scheduling conflicts
- Updates tour status
- Emits WebSocket event

`withdrawFromTour(req, res)`

- Removes guide from tour
- Notifies administrators
- Updates tour status

2.2.3 `individualTourController.js/`

`addIndividualTour(req, res)`

- Creates individual tour request
- Validates date availability
- Sends confirmation email

`approveTour(req, res)`

- Approves individual tour
- Assigns guide
- Generates cancellation token
- Sends approval email

2.2.4 `feedbackController.js/`

`submitFeedback(req, res)`

- Handles feedback submission
- Supports multiple feedback types
- Validates submission token

`getFeedbackByRole(req, res)`

- Retrieves feedback based on user role

- Filters by user type
- Returns formatted feedback data

2.2.5 `userManagementController.js/`

`removeUser(req, res)`

- Deletes user account
- Handles role-specific cleanup
- Removes associated records

`changeUserRole(req, res)`

- Updates user role
- Handles role-specific data
- Updates user type mapping

2.2.6 `workController.js/`

`getAllWorkEntries(req, res)`

- Retrieves all work records
- Combines different work types
- Includes approval status

`saveWorkload(req, res)`

- Updates work entry workload
- Validates input
- Supports multiple work types

2.2.7 `tourLocationController.js/`

`getAllLocations(req, res)`

- Lists all tour locations
- Includes capacity info
- Returns current status

`startTour(req, res)`

- Initiates tour at location

- Updates occupancy
- Validates capacity

2.2.8 `schoolController.js/`

`addSchool(req, res)`

- Creates new school record
- Calculates credit score
- Validates school data

`updateSchool(req, res)`

- Updates school information
- Recalculates credit score
- Maintains historical data

2.2.9 `fairController.js/`

`createFair(req, res)`

- Creates fair event
- Validates date availability
- Handles organization details

`assignGuide(req, res)`

- Assigns guide to fair
- Sends notification emails
- Updates fair status

2.2.10 `profileController.js/`

`getProfile(req, res)`

- Retrieves user profile
- Returns formatted data
- Includes role-specific info

`uploadProfilePicture(req, res)`

- Handles profile image upload

- Processes image data
- Updates profile record

2.2.11 `guideInfoController.js/`

`getGuideInfo(req, res)`

- Retrieves guide information
- Supports filtering and sorting
- Includes schedule data

2.2.12 `advisorController.js/`

`getAdvisors(req, res)`

- Lists all advisors
- Includes candidate guides
- Returns formatted data

`getAdvisorDetails(req, res)`

- Retrieves specific advisor info
- Includes assigned candidates
- Returns complete profile

2.3 `services/`

Contains service layer implementations:

- `EmailService.js` - Email notification handling
- `OtpService.js` - OTP generation and verification
- `TourLocationService.js` - Realtime status tour location tracking
- `UserService.js` - User management operations

2.3.1 `EmailService.js/`

Template-based email service using nodemailer.

`sendRegistrationEmail(email, password)`

Sends account credentials to new users.

- Uses registration HTML template

- Includes login URL from FRONTEND_URL env
- Returns Promise from sendEmail

`sendLoginOTPEmail(email, otp)`

Sends login verification code.

- 5-minute validity period
- Formatted OTP in HTML template
- Returns Promise from sendEmail

`sendPasswordResetEmail(email, resetLink)`

Sends password reset link.

- 1-hour valid reset token
- Includes FRONTEND_URL/reset-password path
- Returns Promise from sendEmail

`sendIndividualTourConfirmationEmail(email, { name, tour_date, time })`

Sends tour booking confirmation.

- Formats date to TR locale
- Returns Promise from sendEmail

Template Methods

Private methods for HTML email templates:

- _getRegistrationTemplate(email, password)
- _getOTPTemplate(otp)
- _getPasswordResetTemplate(resetLink)
- _getFeedbackRequestTemplate(name, tour_date, time, feedback_link)

2.3.2 `OtpService.js/`

OTP operations with PostgreSQL storage.

`generateAndSaveOTP(userId)`

- Generates 6-digit OTP
- Stores in users table with 5-minute expiry
- Updates existing OTP if present
- Returns generated OTP string

`verifyOTP(userId, otp)`

- Verifies against stored OTP
- Checks expiration timestamp
- Clears OTP on successful verification
- Returns boolean

`clearOTP(userId)`

- Nullifies two_factor_secret
- Clears reset_token_expires
- Used after successful verification

2.3.3 `TourLocationService.js/`

Real-time location tracking with PostgreSQL.

`getAllLocations()`

Fetches locations with current status:

- Current occupancy
- Status (empty/partial/full)
- Connected users list
- Real-time capacity tracking

`incrementOccupancy(id, userId)`

Transaction-based occupancy update:

- Checks existing user locations
- Validates against capacity
- Updates user_locations table
- Returns updated location state

`decrementOccupancy(id, userId)`

Reduces location occupancy:

- Validates user presence
- Updates occupancy count
- Removes user_locations record
- Returns updated state

`getLocationWithUsers(id)`

Detailed location query with active users:

- Current occupancy
- User details
- Join timestamps
- Dynamic status calculation

2.3.4 `UserService.js/`

User management with role-specific logic.

`createUser(userData)`

Creates user with role setup:

- Generates random password
- Maps role to userType (1-4)
- Handles role-specific tables
- Returns userId and original password

`_handleRoleSpecificLogic(userId, role, userData)`

Role-based setup operations:

- Advisor: Creates advisor record
- Candidate Guide: Links to advisor
- Updates related tables

Password Operations

Bcrypt-based password handling:

- hashPassword(password): Returns hashed string
- verifyPassword(plain, hashed): Returns boolean
- updateUserPassword(email, hashedNew)

Token Operations

Password reset handling:

- requestPasswordReset(email): Returns reset token
- resetPassword(token, newPassword): Updates password
- Uses JWT with 1-hour expiry

2.4 `utils/`

Utility functions and helpers:

- `jwt.js` - JWT token handling
- `email.js` - Email utilities
- `otp.js` - OTP utilities

2.4.1 `email.js/`

Core email functionality using nodemailer with Gmail SMTP.

`sendEmail({ to, subject, html })`

Sends emails using configured SMTP settings. Requires EMAIL_USER and EMAIL_PASSWORD env variables.

- Parameters object contains recipient, subject line, and HTML content
- Returns Promise resolving to send status
- Throws on SMTP failure or missing config

`sendConfirmationEmail(toEmail, { teacher_name, tour_date, school_name, time_preferences })`

Sends tour confirmation using predefined template.

- Formats date to TR locale
- Includes school details and time preferences
- Uses basic HTML template with tour details
- Calls sendEmail internally

2.4.2 `jwt.js/`

JWT operations using jsonwebtoken library.

`generateToken(userId, user_type, isTemp)`

Creates authentication token.

- userId: user identifier
- user_type: numeric role identifier
- isTemp: boolean for temporary OTP tokens
- 1 hour expiration
- Signs with JWT_SECRET env variable

`verifyToken(token)`

Validates and decodes JWT.

- Returns decoded payload if valid
- Throws on invalid signature or expired token

`generateCancellationToken(tourId, tourDate, type)`

Creates tour cancellation token.

- tourId: tour identifier
- tourDate: used for expiration
- type: "tour" | "individual_tour" | "fair"
- Expires at tour date

`generateFeedbackToken(tourId, type)`

Creates feedback submission token.

- 7 day expiration
- Includes tour identifier and type
- Used for feedback validation

2.4.3 `otp.js/`

OTP generation and validation.

`generateOtp()`

Creates 6-digit numeric OTP.

- Returns string
- Range: 100000-999999
- Used for 2FA

`validateOtp(otp)`

Validates OTP format.

- Returns boolean
- Checks for 6 digits only

`isOtpExpired(expiryTime)`

Checks OTP expiration.

- expiryTime: timestamp
- Returns boolean
- Compares against current time
- Used for 5-minute validity window

3. Frontend Structure

3.1 `src/`

- `assets/`

  Stores static files used throughout the application.

  - Images, icons, SVGs
  - Default profile pictures
  - Global stylesheets
  - Font files
  - Other media resources

- `components/`

  Reusable React components following atomic design pattern.

  - Common UI elements (buttons, inputs, modals)
  - Layout components (sidebar, headers, footers)
  - Form components and validators
  - Data display components (tables, cards)
  - Navigation components
  - Each component may have its own:
    - CSS modules
    - Sub-components

- `context/`

  React Context API implementations for global state management.

  - Authentication context
  - User preferences context
  - Theme context
  - Global notifications
  - Provides state and dispatch functions to child components

- `hooks/`

  Custom React hooks for shared functionality.

  - API data fetching hooks
  - Form handling hooks
  - Authentication hooks
  - Utility hooks (window size, scroll position)
  - State management hooks

- `pages/`

  Top-level page components representing different routes.

  - One component per route
  - Page-specific logic
  - Layout composition
  - Data fetching and state management
  - Each page may have:
    - CSS modules
    - Local components
    - Page-specific hooks

- `routes/`

  Route configuration and navigation management.

  - Route definitions
  - Protected route wrappers
  - Navigation guards
  - Role-based access control
  - Route constants

- `services/`

  API communication layer and business logic.

  - Axios instances and interceptors
  - API endpoint definitions
  - Data transformation
  - Error handling
  - Service-specific utilities
  - WebSocket connections
  - Local storage operations

3.2 services/

- AdvisorService: Advisor data management and operations. Handles CRUD operations for advisors, manages advisor-candidate relationships, and provides methods for advisor day management. Includes functions for fetching assigned candidate guides and calculating advisor workloads.
- ApproveTourService: Tour approval workflow operations. Manages the entire tour approval process including status updates, email notifications, and schedule confirmations. Handles tour time selection and classroom assignments.
- AssignTourService: Tour guide assignment operations. Manages the assignment of guides and candidate guides to tours, handles scheduling conflicts, and maintains guide quotas. Provides real-time updates through WebSocket for tour assignments.
- AuthService: Authentication and authorization operations. Manages user login/logout flow, handles OTP verification, and maintains JWT tokens. Includes role-based access control and session management.
- DataService: General data fetching and manipulation. Provides centralized data operations for statistics, reports, and analytics. Handles data formatting and transformation for various dashboard displays.
- FairService: Fair event management operations. Handles fair event creation, guide assignments, and status management throughout the fair lifecycle. Includes email notifications and attendance tracking.
- FeedbackService: Tour feedback handling operations. Manages feedback collection for tours and fairs, handles feedback token validation, and stores feedback responses. Includes reporting and analysis functions.

- GuideInfoService: Guide information management. Maintains guide profiles, schedules, and availability information. Handles guide status updates and schedule management.
- IndividualTourService: Individual tour request handling. Manages personal tour requests, scheduling, and guide assignments. Includes visitor communication and tour status tracking.
- PasswordService: Password reset and update operations. Handles password reset tokens, manages password change workflows, and enforces password policies. Includes email notifications for password-related actions.
- ProfileSettingsService: User profile management. Handles user profile updates, image uploads, and preference management. Includes IBAN and contact information management.
- UserService: User account operations. Manages user creation, updates, and role management. Handles user type-specific operations and maintains user hierarchies.
- WorkService: Work record management. Tracks work hours, manages workload calculations, and handles work approval workflows. Includes reporting and payment-related calculations.

3.2 pages/

- AdvisorPage: Advisor management and candidate guide assignments. Displays advisor schedules, their assigned candidate guides, and allows management of advisor-candidate relationships. Includes advisor workload monitoring and day management interface.
- CancellationPage: Tour cancellation handling. Processes tour cancellation requests using secure tokens, sends notifications to relevant parties, and updates tour statuses. Includes cancellation reason tracking.
- DashboardPage: Main overview dashboard. Includes the role-specific Sidebar.
- DataInsightPage: Data visualization and analytics. Presents tour statistics, school performance metrics, and guide activity data through charts and tables. Includes filtering and date range selection.
- FairApprovalPage: Fair request approval workflow. Lists pending fair requests, displays detailed fair information, and manages approval/rejection process.
- FairAssignmentPage: Guide assignment for fairs. Manages guide allocation to fairs, shows guide availability, and handles scheduling conflicts. Includes guide preference handling.
- FeedbackPage: Tour feedback submission and review. Displays collected feedback and allows management of it.
- ForgotPassword: Password recovery workflow. Handles email verification, sends reset links, and guides users through recovery process.
- GuideInfoPage: Guide information display. Shows guide profiles, and schedules. Includes work history and availability management.

- `IndividualTourPage:` Individual tour request management. Handles personal tour requests, scheduling, and guide assignments. Includes visitor information management.
- `LoginPage:` User authentication. Handles user login with email/password and initiates OTP verification. Includes forgot password functionality.
- `OtpVerification:` Two-factor authentication. Manages OTP input, verification, and retry logic. Includes countdown timer and resend functionality.
- `PuantajPage:` Work tracking and management. Displays work records, allows work hour submissions, and manages approvals. Includes payment calculation interface.
- `RealtimeStatus:` Real-time tour location tracking. Shows live location of ongoing tours, occupancy status of locations, and guide positions. Includes interactive map interface.
- `ResetPassword:` Password reset workflow. Handles password reset token validation and new password setup. Includes password strength requirements.
- `SchoolFeedbackPage:` School-specific feedback management. Manages feedback collection for school tours and displays historical feedback.
- `SettingsPage:` User preferences and settings. Manages user profile settings, notification preferences, and account configurations. Includes profile picture management.
- `TourApproval:` Tour request approval workflow. Manages incoming tour requests, schedule verification, and approval process. Includes capacity checking and conflict detection.
- `TourAssignment:` Guide assignment for tours. Handles guide allocation to tours, manages schedules, and prevents booking conflicts.
- `Unauthorized:` Access denied page. Shows access restriction information and redirects to the login page.

- UserManagementPage: User account administration. Manages user accounts, roles, and permissions across the system.