Lafefny - Travel Experience Platform

Lafefny is a comprehensive travel experience platform that connects tourists with local tour guides, activities, and curated itineraries across Egypt. The platform facilitates authentic cultural experiences while streamlining the booking and management process for all stakeholders including tourists, tour guides, activity providers, and local sellers.

Motivation

This project was created to address several key challenges in the Egyptian tourism industry:

- Fragmented booking processes for activities and tour guides
- Limited access to authentic local experiences for tourists
- Lack of digital infrastructure for local tourism service providers
- Need for quality control and standardization in tourism services

Lafefny aims to modernize and streamline the tourism experience by:

- Providing a centralized platform for discovering and booking activities
- Enabling direct connections between tourists and verified local guides
- Supporting local businesses through an integrated marketplace
- Implementing a robust review and rating system
- Offering personalized travel recommendations

Build Status

Current Version: 0.0.0

- Known Issues

1) Application performance needs optimization with larger datasets
2) Some responsive design improvements needed for mobile views
3) Caching system needed for better API response times

Code Style

Frontend Code Style:

- React functional components with hooks
- ESLint configuration with recommended rules
- Tailwind CSS for styling following utility-first approach
- File naming convention: PascalCase for components (e.g. Navigation.jsx)
- Component structure:

```
// Imports
```

```
import { useState, useEffect } from 'react'
import { componentName } from './components'

// Component definition
const ComponentName = ({ props }) => {
  // State/hooks
  // Helper functions
  // Return JSX
}

export default ComponentName
```

Backend Code Style:

- Node.js/Express RESTful API conventions
- MongoDB Schema definitions using Mongoose
- Route structure:

```
const express = require('express')
const router = express.Router()

router.get('/', asyncHandler(async (req, res) => {
  // Route logic
}))

module.exports = router
```

General Guidelines:

- 2 space indentation
- Semi-colons required
- Single quotes for strings
- Async/await preferred over promises
- Destructuring when accessing object properties
- Meaningful variable/function names in camelCase
- JSDoc comments for complex functions

Code Formatting The project uses Prettier with the following key settings:

```
{
  "semi": true,
  "singleQuote": true,
  "tabWidth": 2,
  "trailingComma": "es5"
}
```

Tech/Framework Used

Frontend

- React 18.3.1 with Vite
- Tailwind CSS for styling
- Radix UI for accessible components
- Lucide and Shadcn UI for icons
- Axios for HTTP requests
- React Router v6 for routing

Backend

- Node.js with Express
- MongoDB with Mongoose
- JWT for authentication
- Bcrypt for password hashing
- Node-cron for scheduled tasks
- Stripe for payments
- Multer for file uploads
- Nodemailer for emails
- Amadeus API for bookings

DevOps & Tools

- Git for version control
- ESLint for code linting
- Prettier for code formatting
- VS Code as primary IDE

Features

User Management

- Multi-role authentication (Tourist, Tour Guide, Tourism Governer ,Seller, Advertiser, Admin)
- Document verification for service providers
- Profile management with customizable preferences
- Loyalty points system

Travel Experience

- Interactive itinerary planning
- Activity booking and management
- Real-time availability checking
- Flight and hotel booking
- Location-based recommendations
- Virtual tours and previews

Business Tools

- Dashboard for tour guides and sellers
- Analytics and booking reports
- Document upload and verification
- Commission management
- Inventory tracking

Platform Features

- Real-time notifications
- Multi-language support
- Multi-currency support
- Rating and review system
- Secure payment processing
- Admin moderation tools

Code Examples

- Authentication Flow

```
// Frontend: Sign-in component
const handleSubmit = async (e) => {
  e.preventDefault();
  const response = await signIn({
    username: formData.username,
    password: formData.password
  });
  if (response.token) {
    localStorage.setItem('token', response.token);
    navigate('/dashboard');
  }
};
```

- Booking System

```
// Backend: Booking creation
router.post('/bookings', async (req, res) => {
  try {
    const { activityId, userId, date } = req.body;
    const booking = await Booking.create({
      activity: activityId,
      user: userId,
      date: new Date(date),
      status: 'confirmed'
    });
    await updateLoyaltyPoints(userId, 'booking');
    res.status(201).json(booking);
  } catch (error) {
```

```
    res.status(400).json({ error: error.message });
  }
});
```

- Notification System

```
// Service: Birthday promotion
const checkBirthdays = async () => {
  const today = new Date();
  const users = await User.find({
    'profile.birthDate': {
      $month: today.getMonth() + 1,
      $dayOfMonth: today.getDate()
    }
  });

  for (const user of users) {
    await Notification.create({
      recipient: user._id,
      type: 'BIRTHDAY',
      message: `Happy Birthday! Here's a special discount`
    });
  }
};
```

Installation

Prerequisites

- Node.js (v16.0.0 or higher)
- MongoDB (v6.0 or higher)
- Git
- Step 1: Clone Repository

```
git clone https://github.com/your-username/lafefny.git
cd lafefny
```

- Step 2: Backend Setup

```
cd Backend
npm install
```

- Create .env file in Backend directory:

```
MONGODB_URI=your_database_url
JWT_SECRET=your_jwt_secret
STRIPE_SECRET_KEY=your_stripe_key
AMADEUS_API_KEY=your_amadeus_key
EMAIL_SERVICE=gmail
```

```
EMAIL_USER=your_email
EMAIL_PASS=your_app_password
```

- Start MongoDB:

```
# Windows (if installed as a service)
net start MongoDB
```

- Run backend server:

```
npm run dev
```

- Step 3: Frontend Setup

```
cd ../Frontend/Lafefny
npm install
```

- Start development server:

```
npm run dev
```

- Access application at http://localhost:5173

API Reference

- Admin Routes (/admin)

```
DELETE /admin/delete-account/:userId    // Delete user account
POST /admin/add-tourism-governor        // Add new tourism governor
POST /admin/add-admin                   // Add new admin
PUT /admin/accept/:id                   // Accept a user
PUT /admin/reject/:id                   // Reject a user
GET /admin/numberOfUsers                // Get total user count
GET /admin/numberOfNewUsers             // Get new users count by month
GET /admin/sales-report                 // Get sales report
```

- Tourist Routes (/tourist)

```
GET /tourist/test                       // Test route
GET /tourist/getTouristInfo/:id         // Get tourist info
PATCH /tourist/updateTouristInfo/:id    // Update tourist info
PUT /tourist/updatePreferences/:id      // Update preferences
GET /tourist/getTouristPreferences/:id  // Get preferences
POST /tourist/addTourist               // Add new tourist
POST /tourist/:userId/transportation-booking  // Book transportation
GET /tourist/advertisers               // Get accepted advertisers
GET /tourist/:userId/wishlist          // Get wishlist
DELETE /tourist/:userId/wishlist/:productId  // Remove from wishlist
GET /tourist/:userId/cart              // Get cart
PUT /tourist/:userId/cart/:productId   // Update cart item
DELETE /tourist/:userId/cart/:productId // Remove from cart
```

```
POST /tourist/:activityId/pay          // Pay for activity
POST /tourist/:userId/orders           // Create order
GET /tourist/:userId/orders            // Get orders
PUT /tourist/:userId/orders/:orderId/cancel  // Cancel order
POST /tourist/:touristId/add-wallet    // Add to wallet
POST /tourist/:userId/bookmark-activity/:activityId // Bookmark activity
GET /tourist/:userId/bookmarked-activities   // Get bookmarked activities
GET /tourist/:userId/bookmarked-tours  // Get bookmarked tours
```

- Activity Routes (/activities)

```
POST /activities                     // Create activity
GET /activities                      // Get all activities
GET /activities/:id                  // Get activity by ID
PUT /activities/:id                  // Update activity
DELETE /activities/:id               // Delete activity
GET /activities/upcomingActivities  // Get upcoming activities
GET /activities/searchActivities    // Search activities
POST /activities/:id/cancel          // Cancel activity booking
```

- Itinerary Routes (/itineraries)

```
POST /itineraries                    // Create itinerary
GET /itineraries                     // Get all itineraries
GET /itineraries/:id                // Get itinerary by ID
PUT /itineraries/:id                // Update itinerary
DELETE /itineraries/:id             // Delete itinerary
GET /itineraries/upcomingItineraries // Get upcoming itineraries
GET /itineraries/filterItineraries // Filter itineraries
GET /itineraries/searchItineraries // Search itineraries
GET /itineraries/:id/availableDates // Get available dates
POST /itineraries/:id/book          // Book itinerary
GET /itineraries/tourGuide/:id     // Get tour guide's itineraries
```

- Museum Routes (/museums)

```
POST /museums                        // Create museum
GET /museums                         // Get all museums
GET /museums/:id                     // Get museum by ID
PUT /museums/:id                     // Update museum
DELETE /museums/:id                  // Delete museum
GET /museums/upcomingMuseumEvent   // Get upcoming museum events
GET /museums/filterMuseums          // Filter museums
GET /museums/searchMuseums          // Search museums
```

- Tour Guide Routes (/tourGuide)

```
POST /tourGuide/addTourGuideInfo/:id   // Add tour guide info
GET /tourGuide/getTourGuide/:id        // Get tour guide info
PATCH /tourGuide/updateTourGuideInfo/:id // Update tour guide info
```

```
PATCH /tourGuide/uploadPicture/:id     // Upload picture
PATCH /tourGuide/uploadPDF/:id         // Upload PDF
GET /tourGuide/getPDF/:id              // Get PDF
GET /tourGuide/numberOfTourists/:id     // Get number of tourists
GET /tourGuide/tourguide/sales-report   // Get sales report
```

- Seller Routes (/seller)

```
POST /seller/addSellerInfo/:id         // Add seller info
GET /seller/getSeller/:id              // Get seller info
PATCH /seller/updateSellerInfo/:id     // Update seller info
PATCH /seller/uploadLogo/:id           // Upload logo
PATCH /seller/uploadPDF/:id            // Upload PDF
GET /seller/getPDF/:id                 // Get PDF
GET /seller/sales-report/:userId       // Get sales report
```

- Amadeus Routes (/amadeus)

```
GET /amadeus/iata-code                // Get IATA code
GET /amadeus/search-flights           // Search flights
GET /amadeus/flight-price-offers      // Get flight price offers
POST /amadeus/book-flight             // Book flight
POST /amadeus/hotelOffer             // Search hotel offers
PUT /amadeus/bookHotels              // Book hotels
```

- Product Routes (/products)

```
GET /products                      // Get all products
POST /products                     // Create product
GET /products/:id                  // Get product by ID
PUT /products/:id                  // Update product
DELETE /products/:id               // Delete product
GET /products/filter               // Filter products
POST /products/:id/reviews         // Add product review
GET /products/check-purchase/:userId/:productId  // Check if product purchased
PATCH /products/:id/toggleArchive  // Toggle product archive status
```

- Notification Routes (/notifications)

```
GET /notifications/user/:userId    // Get user notifications
PUT /notifications/:id/read        // Mark notification as read
DELETE /notifications/:id          // Delete notification
```

- Tourist Itinerary Routes (/touristItinerary)

```
POST /touristItinerary            // Create tourist itinerary
GET /touristItinerary            // Get all tourist itineraries
GET /touristItinerary/:id        // Get specific tourist itinerary
PUT /touristItinerary/:id        // Update tourist itinerary
DELETE /touristItinerary/:id     // Delete tourist itinerary
POST /touristItinerary/:id/pay   // Process payment for itinerary
```

- Complaint Routes (/complaints)

```
POST /complaints                  // Submit complaint
GET /complaints                  // Get all complaints
GET /complaints/:id              // Get specific complaint
PUT /complaints/:id             // Update complaint status
DELETE /complaints/:id          // Delete complaint
```

- Password Reset Routes (/forget)

```
POST /forget/password           // Request password reset
POST /forget/reset/:token      // Reset password with token
GET /forget/verify/:token      // Verify reset token
```

- Activity Category Routes (/activityCategory)

```
POST /activityCategory          // Create category
GET /activityCategory          // Get all categories
PUT /activityCategory/:id     // Update category
DELETE /activityCategory/:id  // Delete category
```

- Preference Tag Routes (/preferenceTag)

```
POST /preferenceTag           // Create preference tag
GET /preferenceTag           // Get all preference tags
PUT /preferenceTag/:id       // Update preference tag
DELETE /preferenceTag/:id    // Delete preference tag
```

- Museum Tag Routes (/museumTags)

```
POST /museumTags             // Create museum tag
GET /museumTags             // Get all museum tags
PUT /museumTags/:id         // Update museum tag
DELETE /museumTags/:id      // Delete museum tag
```

- Promo Routes (/promos)

```
POST /promos                  // Create promo code
GET /promos                  // Get all promo codes
GET /promos/:code           // Validate promo code
PUT /promos/:id            // Update promo code
DELETE /promos/:id         // Delete promo code
```

- User Routes

```
// Authentication
POST /signup                     // Register new user
POST /signin                     // User login
GET /profile                     // Get user profile (protected)

// User Management
GET /users                       // Get all users
```

```
GET /users/:id                   // Get user by ID
PUT /change-password/:id         // Change user password
DELETE /delete-account/:userId   // Delete user account
DELETE /seller-delete/:id        // Delete seller account

// Account Deletion Requests
PUT /request-deletion/:userId    // Request account deletion
PUT /cancel-deletion/:userId     // Cancel deletion request

// Admin User Management
PUT /accept/:id                   // Accept user registration
PUT /reject/:id                  // Reject user registration

// Admin Creation
POST /add-admin                 // Add new admin
```

How to Use Lafefny

User Registeration & Login:

- For Tourists

1. Navigate to http://localhost:5173/sign
2. Choose "Tourist" role
3. Fill in registration details
4. Log in with credentials

- For Service Providers (Tour Guides, Sellers, Advertisers)

1. Navigate to http://localhost:5173/sign
2. Select appropriate role
3. Complete registration form
4. Upload required verification documents (PDF format)
5. Wait for admin approval (typically 24-48 hours)

Tourist Features:

- Browse Activities

1. Go to Activities page
2. Use filters for:
    - Tags
    - Price range
    - Ratings
3. Click on activity card for details
4. Book directly or add to bookmarks

- Browse Itinerary

1. Navigate to "Tours"
2. Use Filter or Search options for:
    - Prefered language
    - Tags
    - Price range
3. Click on itinerary card for details
4. Book directly or add to bookmarks

Tour Guide Features:

- Manage Profile

1. Go to Dashboard
2. Update profile information:
    - Bio
    - Experience
    - Languages
    - Availability calendar
3. Upload certificates/documents

- Handle Bookings

1. Check "Upcoming Tours" section
2. Accept/decline new requests
3. View tourist details
4. Message tourists
5. Update tour status

Payment Processing:

- Add Payment Method

1. Go to Account Settings
2. Select "Payment Methods"
3. Add credit card or wallet
4. Set default payment method

- Make Payment

1. Select activity/tour
2. Review booking details
3. Apply loyalty points (optional)
4. Choose payment method
5. Confirm payment
6. Receive confirmation email

Admin Panel:

- Access Admin Dashboard

1. Login with admin credentials
2. Navigate to admin dashboard
3. View system statistics
4. Manage user accounts
5. Handle verification requests

Contribute

Lafefny stands out through its intuitive design philosophy that prioritizes user experience without sacrificing functionality using visual feedback, clear navigation paths, and intuitive workflows that make travel planning accessible to users of all technical backgrounds. Addtionally, our loyalty points system seamlessly integrates with every transaction, automatically rewarding users without requiring manual input and reducing planning time from hours to minutes.

Credits

- Mohanad49
- youssefsameh55
- ZiaddAmgad
- Abde1rahmanEMad
- youssefa460
- Ziadwess2003
- modyyy11
- Kimo1312
- salama-02
- mohdreda2003