

GlobeTrotter - Complete MERN Stack Project Structure

Technology Stack

Frontend: React.js with React Router, Axios, Material-UI/Tailwind CSS

Backend: Node.js with Express.js

Database: MongoDB Atlas (Cloud Database)

Authentication: JWT-based with bcrypt

File Storage: Cloudinary for images and documents

External APIs:

- REST Countries API (Free) for country/city data
- OpenWeatherMap API (Free tier) for weather info
- Unsplash API (Free) for destination photos
- Exchange Rates API (Free) for currency conversion
- Google Places API (has free tier) for location suggestions

Email Service: SendGrid/Nodemailer for trip sharing notifications

Deployment: Frontend (Vercel/Netlify), Backend (Railway/Render)

Complete Page Structure

Authentication Pages

1. Landing Page (/)

Components: Hero section, features showcase, testimonials, CTA buttons

Functionalities:

- Welcome banner with travel imagery
- Feature highlights (multi-city planning, budget tracking, sharing)
- Sample itinerary previews
- Navigation to login/register
- Footer with links and contact info
- Responsive design for mobile/desktop

2. Login Page (/login)

Components: Login form, social login options

Functionalities:

- Email and password fields
- JWT token generation and storage in localStorage
- "Remember me" checkbox
- "Forgot Password" link
- Google OAuth integration

- Redirect to dashboard after login
- Form validation with error handling

3. Register Page (/register)

Components: Registration form with email verification

Functionalities:

- Name, email, password, confirm password fields
- Profile picture upload to Cloudinary
- Email verification via OTP
- Travel preferences setup (optional)
- Terms and conditions acceptance
- Real-time validation
- Password strength meter

4. Forgot Password (/forgot-password)

Components: Email input, OTP verification, password reset

Functionalities:

- Email input for password reset
- OTP verification via email
- New password setup with validation
- Success confirmation
- Redirect to login

User Dashboard & Trip Management

5. Dashboard / Home Screen (/dashboard)

Components: Welcome widget, trip cards, quick actions, travel inspiration

Functionalities:

- Personalized welcome message
- Recent trips overview (3-4 cards)
- "Plan New Trip" prominent button
- Upcoming trip alerts
- Travel statistics (total trips, countries visited)
- Recommended destinations based on preferences
- Quick access to shared trips
- Weather widget for upcoming trips

6. My Trips (/trips)

Components: Trip grid/list, search bar, filters, pagination

Functionalities:

- All trips display in card/list format
- Search trips by name, destination
- Filter by: Date, status (planned/completed), duration
- Sort by: Date created, departure date, alphabetical
- Trip cards show: name, dates, destination count, budget
- Quick actions: Edit, Delete, Share, Duplicate
- Infinite scroll or pagination
- Export trip data option

7. Create Trip (/trips/create)

Components: Multi-step trip creation form

Functionalities:

- **Step 1:** Basic info (trip name, description, dates)
- **Step 2:** Cover photo upload (Cloudinary)
- **Step 3:** Initial destination selection
- **Step 4:** Budget range setting
- **Step 5:** Privacy settings (private/public)
- Save as draft option
- Progress indicator
- Form validation at each step

Trip Planning & Itinerary

8. Trip Details (/trips/:id)

Components: Trip header, itinerary timeline, sidebar actions

Functionalities:

- Trip overview with cover photo
- Dates, duration, total budget display
- Itinerary timeline view
- Quick stats (cities, activities, cost per day)
- Action buttons: Edit, Share, Duplicate, Delete
- Weather forecast for destinations

- Currency converter widget
- Export to PDF option

9. Itinerary Builder (/trips/:id/edit)

Components: Drag-drop interface, city selector, activity planner

Functionalities:

- Interactive timeline builder
- Drag and drop to reorder cities/activities
- "Add Stop" with city search integration
- Date assignment for each city
- Activity scheduling with time slots
- Budget allocation per city/activity
- Real-time cost calculation
- Save draft changes
- Preview mode toggle

10. Itinerary View (/trips/:id/itinerary)

Components: Day-wise timeline, activity cards, cost breakdown

Functionalities:

- Clean day-by-day layout
- City headers with arrival/departure
- Activity blocks with time, cost, description
- View modes: Timeline, Calendar, List
- Print-friendly format
- Share specific days
- Notes section for each day
- Weather information integration

Search & Discovery

11. City Search (/cities)

Components: Search interface, city cards, filters, map integration

Functionalities:

- Global city search with autocomplete
- Filter by: Country, continent, climate, cost index
- City cards with: photo, country, cost level, popularity

- "Add to Trip" quick action
- Detailed city information popup
- Map view toggle
- Save favorite cities
- Recently searched cities

12. Activity Search (/activities)

Components: Activity browser, category filters, cost filters

Functionalities:

- Browse activities by category (food, adventure, culture, etc.)
- Filter by: Type, duration, cost range, rating
- Activity cards with: photo, description, estimated cost, duration
- Add to specific trip days
- Activity reviews and ratings
- Map location display
- Save favorite activities
- Integration with local attractions API

13. Single City Details (/cities/:id)

Components: City overview, attractions, weather, cost info

Functionalities:

- City photo gallery
- General information and highlights
- Top attractions and activities
- Current weather and climate info
- Cost of living indicators
- Best time to visit
- Popular itineraries featuring this city
- "Add to Trip" with date selection



Budget & Analytics

14. Trip Budget (/trips/:id/budget)

Components: Budget breakdown, expense tracker, cost analysis

Functionalities:

- Total budget overview with remaining amount

- Category-wise breakdown (transport, accommodation, food, activities)
- Cost per day analysis
- Budget vs actual expense tracking
- Currency conversion for international trips
- Budget alerts and warnings
- Export expense report
- Visual charts (pie, bar, line)

15. Budget Calculator (/budget-calculator)

Components: Trip cost estimator, comparison tool

Functionalities:

- Estimate trip cost based on destinations
- Duration and travel style inputs
- Real-time cost calculation
- Compare costs between different cities
- Budget recommendations based on trip type
- Save estimates for future reference
- Share cost estimates

Social Features

16. Shared Trips (/explore)

Components: Public trip gallery, search, featured trips

Functionalities:

- Browse public trips from other users
- Search by destination, duration, budget range
- Featured trips by community/admin
- Trip previews with basic info
- "Copy Trip" functionality
- Like and save public trips
- Filter by trip type or theme
- User profiles of trip creators

17. Public Trip View (/trips/:id/public)

Components: Read-only trip display, copy options

Functionalities:

- Clean, shareable trip view
- Complete itinerary display
- Budget overview (if shared)
- "Copy This Trip" button
- Social sharing buttons
- Trip creator profile link
- Similar trip suggestions
- Print/PDF export option

18. Trip Sharing (/trips/:id/share)

Components: Sharing options, privacy settings, collaborative features

Functionalities:

- Generate shareable link
- Privacy level selection (public, link-only, private)
- Email sharing with custom message
- Social media sharing integration
- Collaborative editing permissions
- QR code generation for mobile sharing
- View sharing analytics (views, copies)

Profile & Settings

19. User Profile (/profile)

Components: Profile info, travel preferences, trip statistics

Functionalities:

- Personal information editing
- Profile picture management
- Travel preferences (budget range, trip types)
- Travel statistics (trips completed, countries visited)
- Privacy settings
- Account security settings
- Connected social accounts
- Delete account option

20. Settings (/settings)

Components: App preferences, notifications, privacy

Functionalities:

- Notification preferences (email, in-app)
- Default currency selection
- Language preferences
- Privacy settings (profile visibility)
- Data export options
- Theme selection (light/dark)
- Time zone settings
- Integration settings



Calendar & Timeline

21. Trip Calendar (/trips/:id/calendar)

Components: Calendar view, timeline, scheduling tools

Functionalities:

- Monthly calendar view of trip
- Day-wise activity overview
- Drag and drop rescheduling
- Conflict detection (overlapping activities)
- Integration with external calendars
- Reminder settings
- Weather overlay on calendar
- Print calendar view



Analytics Dashboard (Optional)

22. Travel Analytics (/analytics)

Components: Personal travel insights, trends, achievements

Functionalities:

- Travel patterns and trends
- Budget analysis over time
- Most visited destinations
- Travel frequency insights
- Carbon footprint calculator
- Travel achievements/badges

- Year in review summary
- Export travel data

🗄 MongoDB Atlas Collections Structure

Users Collection

```
{
  _id: ObjectId,
  email: String, // unique
  password: String, // hashed with bcrypt
  profile: {
    name: String,
    bio: String,
    avatar_url: String, // Cloudinary URL
    location: String,
    travel_preferences: {
      budget_range: String, // 'budget', 'mid-range', 'luxury'
      trip_types: [String], // 'adventure', 'cultural', 'relaxation', etc.
      preferred_duration: String,
      favorite_activities: [String]
    }
  },
  stats: {
    total_trips: Number,
    countries_visited: [String],
    cities_visited: [String],
    total_budget_spent: Number
  },
  settings: {
    currency: String,
    timezone: String,
    notifications: {
      email: Boolean,
```

```
    trip_reminders: Boolean,
    sharing_updates: Boolean
  },
  privacy: {
    profile_public: Boolean,
    trips_public_default: Boolean
  }
},
is_verified: Boolean,
is_active: Boolean,
last_login: Date,
created_at: Date,
updated_at: Date
}
```

Trips Collection

```
{
  _id: ObjectId,
  user_id: ObjectId,
  name: String,
  description: String,
  cover_image: String, // Cloudinary URL
  start_date: Date,
  end_date: Date,
  duration_days: Number,
  status: String, // 'planning', 'confirmed', 'completed', 'cancelled'
  privacy: String, // 'private', 'public', 'link_only'
  budget: {
    total_budget: Number,
    currency: String,
    spent_amount: Number,
    categories: {
```

```
    accommodation: Number,
    transport: Number,
    food: Number,
    activities: Number,
    shopping: Number,
    miscellaneous: Number
  }
},
destinations: [ObjectId], // Reference to destinations
sharing: {
  public_link: String,
  is_shareable: Boolean,
  shared_with: [ObjectId], // User IDs with collaborative access
  view_count: Number,
  copy_count: Number
},
metadata: {
  total_activities: Number,
  total_cities: Number,
  estimated_cost: Number
},
created_at: Date,
updated_at: Date
}
```

Destinations Collection

```
{
  _id: ObjectId,
  trip_id: ObjectId,
  city: {
    name: String,
    country: String,
```

```
country_code: String,
coordinates: {
  lat: Number,
  lng: Number
},
timezone: String,
currency: String
},
arrival_date: Date,
departure_date: Date,
duration_days: Number,
order: Number, // Sequence in trip
accommodation: {
  name: String,
  type: String,
  cost_per_night: Number,
  nights: Number,
  total_cost: Number,
  booking_link: String
},
transport: {
  arrival_method: String,
  departure_method: String,
  arrival_cost: Number,
  departure_cost: Number
},
activities: [ObjectId], // Reference to activities
notes: String,
budget_allocated: Number,
budget_spent: Number
}
```

Activities Collection

```
{
  _id: ObjectId,
  destination_id: ObjectId,
  name: String,
  description: String,
  category: String, // 'food', 'sightseeing', 'adventure', 'culture', etc.
  date: Date,
  start_time: String,
  end_time: String,
  duration_hours: Number,
  cost: Number,
  currency: String,
  location: {
    name: String,
    address: String,
    coordinates: {
      lat: Number,
      lng: Number
    }
  },
  booking_info: {
    booking_required: Boolean,
    booking_link: String,
    contact_info: String
  },
  images: [String], // Cloudinary URLs
  notes: String,
  priority: String, // 'must-do', 'nice-to-have', 'optional'
  created_at: Date,
  updated_at: Date
}
```

```
}
```

Cities Master Collection

```
{
```

```
  _id: ObjectId,
```

```
  name: String,
```

```
  country: String,
```

```
  country_code: String,
```

```
  continent: String,
```

```
  coordinates: {
```

```
    lat: Number,
```

```
    lng: Number
```

```
  },
```

```
  timezone: String,
```

```
  currency: String,
```

```
  cost_index: Number, // 1-5 scale
```

```
  popularity_score: Number,
```

```
  climate: String,
```

```
  best_months: [Number], // Array of month numbers
```

```
  highlights: [String],
```

```
  images: [String], // Curated city images
```

```
  average_costs: {
```

```
    accommodation_budget: Number,
```

```
    accommodation_mid: Number,
```

```
    accommodation_luxury: Number,
```

```
    food_budget: Number,
```

```
    food_mid: Number,
```

```
    food_luxury: Number,
```

```
    transport_daily: Number
```

```
  },
```

```
  created_at: Date,
```

```
  updated_at: Date
```

```
}
```

Shared Trips Collection

```
{
```

```
  _id: ObjectId,
```

```
  trip_id: ObjectId,
```

```
  shared_by: ObjectId,
```

```
  share_link: String,
```

```
  privacy_level: String, // 'public', 'link_only'
```

```
  share_settings: {
```

```
    show_budget: Boolean,
```

```
    allow_copy: Boolean,
```

```
    allow_comments: Boolean
```

```
  },
```

```
  analytics: {
```

```
    views: Number,
```

```
    copies: Number,
```

```
    shares: Number
```

```
  },
```

```
  featured: Boolean,
```

```
  featured_date: Date,
```

```
  created_at: Date,
```

```
  updated_at: Date
```

```
}
```

Notifications Collection

```
{
```

```
  _id: ObjectId,
```

```
  user_id: ObjectId,
```

```
  type: String, // 'trip_reminder', 'sharing', 'budget_alert', 'weather_update'
```

```
  title: String,
```

```
  message: String,
```

```
  data: Object, // Additional context data
```

```
is_read: Boolean,  
created_at: Date  
}
```

API Endpoints Structure

Authentication APIs

- POST /api/auth/register - User registration
- POST /api/auth/login - User login
- POST /api/auth/verify-email - Email verification
- POST /api/auth/forgot-password - Password reset request
- POST /api/auth/reset-password - Password reset confirmation
- POST /api/auth/refresh-token - Refresh JWT token
- POST /api/auth/logout - User logout
- POST /api/auth/google - Google OAuth login

User Profile APIs

- GET /api/user/profile - Get user profile
- PUT /api/user/profile - Update profile
- POST /api/user/upload-avatar - Upload profile picture
- GET /api/user/stats - Get travel statistics
- PUT /api/user/settings - Update user settings
- DELETE /api/user/account - Delete account

Trip Management APIs

- GET /api/trips - Get user trips with pagination
- POST /api/trips - Create new trip
- GET /api/trips/:id - Get trip details
- PUT /api/trips/:id - Update trip
- DELETE /api/trips/:id - Delete trip
- POST /api/trips/:id/duplicate - Duplicate trip
- GET /api/trips/:id/budget - Get trip budget details
- PUT /api/trips/:id/budget - Update trip budget

Destination APIs

- POST /api/trips/:id/destinations - Add destination to trip

- PUT /api/destinations/:id - Update destination
- DELETE /api/destinations/:id - Remove destination
- PUT /api/destinations/:id/reorder - Reorder destinations

Activity APIs

- POST /api/destinations/:id/activities - Add activity
- PUT /api/activities/:id - Update activity
- DELETE /api/activities/:id - Delete activity
- PUT /api/activities/:id/reorder - Reorder activities

Search & Discovery APIs

- GET /api/cities/search - Search cities
- GET /api/cities/:id - Get city details
- GET /api/cities/popular - Get popular cities
- GET /api/cities/recommendations - Get recommended cities
- GET /api/activities/search - Search activities
- GET /api/activities/categories - Get activity categories

Sharing APIs

- POST /api/trips/:id/share - Create sharable link
- GET /api/trips/public/:shareId - Get public trip
- POST /api/trips/:id/copy - Copy public trip
- PUT /api/trips/:id/share-settings - Update sharing settings
- GET /api/explore - Get public trips
- POST /api/trips/:id/like - Like public trip

External API Integration

- GET /api/external/weather/:city - Get weather data
- GET /api/external/exchange-rates - Get currency rates
- GET /api/external/places/:query - Search places
- GET /api/external/country-info/:code - Get country information

Notification APIs

- GET /api/notifications - Get user notifications
- PUT /api/notifications/:id/read - Mark as read
- DELETE /api/notifications/:id - Delete notification

- POST /api/notifications/mark-all-read - Mark all as read

Analytics APIs

- GET /api/analytics/user-stats - Get user travel analytics
- GET /api/analytics/trip-trends - Get trip trends
- GET /api/analytics/budget-analysis - Get budget analysis

Frontend Component Structure

Shared Components

src/

```

├── components/
|   ├── shared/
|   |   ├── Header.jsx
|   |   ├── Footer.jsx
|   |   ├── Sidebar.jsx
|   |   ├── LoadingSpinner.jsx
|   |   ├── ErrorBoundary.jsx
|   |   ├── Modal.jsx
|   |   ├── Toast.jsx
|   |   ├── ProtectedRoute.jsx
|   |   ├── ImageUpload.jsx
|   |   └── ConfirmDialog.jsx
|   ├── forms/
|   |   ├── LoginForm.jsx
|   |   ├── RegisterForm.jsx
|   |   ├── TripForm.jsx
|   |   ├── DestinationForm.jsx
|   |   ├── ActivityForm.jsx
|   |   └── ProfileForm.jsx
|   └── ui/
|       ├── Button.jsx
|       └── Input.jsx

```

- | | └─ Select.jsx
- | | └─ DatePicker.jsx
- | | └─ Card.jsx
- | | └─ Table.jsx
- | | └─ Timeline.jsx
- | | └─ Calendar.jsx
- | | └─ Charts.jsx
- | └─ layout/
 - └─ MainLayout.jsx
 - └─ AuthLayout.jsx
 - └─ PublicLayout.jsx

Trip Management Components

src/

- └─ components/
 - └─ trips/
 - └─ Dashboard/
 - └─ TripDashboard.jsx
 - └─ TripCard.jsx
 - └─ StatsWidget.jsx
 - └─ QuickActions.jsx
 - └─ TripList/
 - └─ TripList.jsx
 - └─ TripGrid.jsx
 - └─ TripFilters.jsx
 - └─ TripSearch.jsx
 - └─ TripBuilder/
 - └─ CreateTrip.jsx
 - └─ TripWizard.jsx
 - └─ BasicInfo.jsx
 - └─ DateSelection.jsx

- | | | └─ CoverPhoto.jsx
- | | └─ Itinerary/
 - | | | └─ ItineraryBuilder.jsx
 - | | | └─ ItineraryView.jsx
 - | | | └─ TimelineView.jsx
 - | | | └─ CalendarView.jsx
 - | | | └─ DayCard.jsx
 - | | | └─ DestinationCard.jsx
 - | | | └─ ActivityCard.jsx
- | | └─ Budget/
 - | | | └─ BudgetOverview.jsx
 - | | | └─ BudgetBreakdown.jsx
 - | | | └─ ExpenseTracker.jsx
 - | | | └─ BudgetCharts.jsx
 - | | | └─ CurrencyConverter.jsx
- | | └─ Sharing/
 - | | | └─ ShareTrip.jsx
 - | | | └─ ShareSettings.jsx
 - | | | └─ PublicView.jsx
 - | | | └─ ShareAnalytics.jsx

Search & Discovery Components

src/

- └─ components/
 - | └─ search/
 - | | └─ CitySearch/
 - | | | └─ CitySearchPage.jsx
 - | | | └─ CityCard.jsx
 - | | | └─ CityDetails.jsx
 - | | | └─ CityFilters.jsx
 - | | | └─ CityMap.jsx

```
| | └─ ActivitySearch/
| | | └─ ActivityBrowser.jsx
| | | └─ ActivityCard.jsx
| | | └─ ActivityFilters.jsx
| | | └─ ActivityDetails.jsx
| | | └─ CategoryTabs.jsx
| | └─ Explore/
| |   └─ PublicTrips.jsx
| |   └─ FeaturedTrips.jsx
| |   └─ TrendingDestinations.jsx
| |   └─ TripPreview.jsx
```

📁 Backend Structure (Node.js/Express)

Main Server Structure

```
backend/
├─ server.js
├─ app.js
├─ config/
| └─ database.js
| └─ cloudinary.js
| └─ email.js
| └─ passport.js
| └─ constants.js
├─ controllers/
| └─ authController.js
| └─ userController.js
| └─ tripController.js
| └─ destinationController.js
| └─ activityController.js
| └─ searchController.js
| └─ sharingController.js
```

- | └─ budgetController.js
- | └─ uploadController.js
- | └─ externalAPIController.js
- | └─ notificationController.js
- └─ models/
 - | └─ User.js
 - | └─ Trip.js
 - | └─ Destination.js
 - | └─ Activity.js
 - | └─ City.js
 - | └─ SharedTrip.js
 - | └─ Notification.js
- └─ routes/
 - | └─ auth.js
 - | └─ users.js
 - | └─ trips.js
 - | └─ destinations.js
 - | └─ activities.js
 - | └─ search.js
 - | └─ sharing.js
 - | └─ external.js
 - | └─ upload.js
 - | └─ notifications.js
- └─ middleware/
 - | └─ auth.js
 - | └─ validation.js
 - | └─ errorHandler.js
 - | └─ fileUpload.js
 - | └─ rateLimiting.js
 - | └─ cors.js

```
└─ utils/
  │ └─ tokenUtils.js
  │ └─ emailUtils.js
  │ └─ imageUtils.js
  │ └─ dateUtils.js
  │ └─ budgetUtils.js
  │ └─ externalAPIs.js
  └─ helpers.js
└─ services/
  │ └─ emailService.js
  │ └─ notificationService.js
  │ └─ weatherService.js
  │ └─ currencyService.js
  └─ geocodingService.js
└─ package.json
```

External API Integration

Free APIs to Use

1. REST Countries API (Free)

// Get country information

GET <https://restcountries.com/v3.1/all>

GET <https://restcountries.com/v3.1/name/{country}>

GET <https://restcountries.com/v3.1/alpha/{code}>

2. OpenWeatherMap API (Free tier: 1000 calls/day)

// Get weather data

GET https://api.openweathermap.org/data/2.5/weather?q={city}&appid={API_key}

GET https://api.openweathermap.org/data/2.5/forecast?q={city}&appid={API_key}

3. ExchangeRate-API (Free: 1500 requests/month)

// Get exchange rates

GET https://api.exchangerate-api.com/v4/latest/{base_currency}

4. Unsplash API (Free: 50 requests/hour)

```
// Get destination photos
```

```
GET https://api.unsplash.com/search/photos?query={destination}&client_id={API_key}
```

5. GeoNames API (Free with registration)

```
// Get city data and coordinates
```

```
GET http://api.geonames.org/searchJSON?q={city}&maxRows=10&username={username}
```

6. Nominatim API (Free - OpenStreetMap)

```
// Geocoding service
```

```
GET https://nominatim.openstreetmap.org/search?format=json&q={location}
```

API Integration Service Example

```
// services/externalAPIs.js
```

```
class ExternalAPIService {
```

```
  // Get weather data
```

```
  async getWeather(city) {
```

```
    const response = await axios.get(
```

```
    `https://api.openweathermap.org/data/2.5/weather?q=${city}&appid=${process.env.WEATHER_API_KEY}`
```

```
    );
```

```
    return response.data;
```

```
  }
```

```
// Get currency exchange rates
```

```
async getExchangeRates(baseCurrency = 'USD') {
```

```
  const response = await axios.get(
```

```
    `https://api.exchangerate-api.com/v4/latest/${baseCurrency}`
```

```
  );
```

```
  return response.data.rates;
```

```
}
```

```
// Get destination photos
```

```
async getDestinationPhotos(destination) {
```

```
  const response = await axios.get(
```



```

    `https://api.unsplash.com/search/photos?query=${destination}&per_page=5`,
    {
      headers: {
        Authorization: `Client-ID ${process.env.UNSPLASH_ACCESS_KEY}`
      }
    }
  );
  return response.data.results;
}

// Get country information
async getCountryInfo(countryCode) {
  const response = await axios.get(
    `https://restcountries.com/v3.1/alpha/${countryCode}`
  );
  return response.data[0];
}
}

```

Deployment Strategy

Frontend (React.js)

Platform: Vercel or Netlify

Build Command: npm run build

Environment Variables:

- REACT_APP_API_URL
- REACT_APP_GOOGLE_OAUTH_CLIENT_ID
- REACT_APP_CLOUDINARY_CLOUD_NAME

Backend (Node.js/Express)

Platform: Railway, Render, or Heroku

Environment Variables:

- MONGODB_URI (MongoDB Atlas connection)
- JWT_SECRET
- JWT_REFRESH_SECRET

- CLOUDINARY_CLOUD_NAME
- CLOUDINARY_API_KEY
- CLOUDINARY_API_SECRET
- EMAIL_SERVICE (SendGrid/Nodemailer config)
- WEATHER_API_KEY
- UNSPLASH_ACCESS_KEY
- GOOGLE_OAUTH_CLIENT_SECRET

Database

MongoDB Atlas: Cloud-hosted MongoDB

Features:

- Automated backups
- Auto-scaling based on usage
- Global clusters for better performance
- Built-in security features



Performance Optimization

Frontend

- **Code Splitting:** React.lazy() for route-based splitting
- **Image Optimization:** Cloudinary transformations and lazy loading
- **State Management:** Context API + useReducer for complex state
- **Caching:** React Query for API call caching
- **Bundle Optimization:** Webpack bundle analyzer
- **PWA Features:** Service workers for offline capabilities

Backend

- **API Caching:** Redis for frequently accessed data
- **Database Optimization:** Proper indexing on search fields
- **Compression:** Gzip compression for API responses
- **Rate Limiting:** Prevent API abuse
- **Connection Pooling:** MongoDB connection optimization
- **CDN:** Cloudinary CDN for image delivery

Database

- **Indexing Strategy:**

- Compound index on user_id + created_at for trip queries
- Text index on city names for search
- Geospatial index for location-based queries
- **Aggregation Optimization:** Use MongoDB aggregation pipeline for complex queries
- **Data Archiving:** Archive completed old trips

Security Considerations

Authentication & Authorization

- JWT with refresh token strategy
- Password hashing with bcrypt (salt rounds: 12)
- Email verification for new accounts
- Rate limiting on auth endpoints
- Google OAuth 2.0 integration

Data Protection

- Input validation and sanitization
- SQL injection prevention (using Mongoose)
- XSS protection with helmet.js
- CORS configuration
- File upload restrictions and validation
- Environment variable protection

Privacy Features

- Trip privacy controls (private/public/link-only)
- User data export functionality
- Account deletion with data cleanup
- Anonymous usage analytics only

Testing Strategy

Frontend Testing

- **Unit Tests:** Jest + React Testing Library
- **Component Testing:** Storybook for UI components
- **Integration Tests:** Cypress for E2E testing
- **Performance Tests:** Lighthouse CI
- **Accessibility Tests:** axe-core integration

Backend Testing

- **Unit Tests:** Jest + Supertest
- **API Testing:** Postman collections
- **Database Testing:** MongoDB Memory Server
- **Load Testing:** Artillery.io
- **Security Testing:** OWASP ZAP

Test Coverage Goals

- Frontend: >80% component coverage
- Backend: >90% route and controller coverage
- Critical user flows: 100% E2E coverage

Development Phases

Phase 1: Foundation (Week 1-2)

Priority: High

- ☐ Project setup and basic structure
- ☐ Authentication system (login/register/JWT)
- ☐ User profile management
- ☐ Basic dashboard layout
- ☐ MongoDB Atlas setup and basic collections
- ☐ Cloudinary integration for image uploads

Phase 2: Core Trip Management (Week 3-4)

Priority: High

- ☐ Trip CRUD operations
- ☐ Basic itinerary builder
- ☐ Destination management
- ☐ Activity management
- ☐ Simple budget tracking
- ☐ Trip list and details views

Phase 3: Search & Discovery (Week 5-6)

Priority: High

- ☐ City search with external API integration
- ☐ Activity search and categories

- ☐ City master data population
- ☐ Basic filters and sorting
- ☐ Integration with weather API
- ☐ Currency conversion features

Phase 4: Enhanced Features (Week 7-8)

Priority: Medium

- ☐ Advanced itinerary builder with drag-drop
- ☐ Calendar and timeline views
- ☐ Budget breakdown and analytics
- ☐ Trip sharing functionality
- ☐ Public trip exploration
- ☐ Notification system

Phase 5: Polish & Optimization (Week 9-10)

Priority: Medium

- ☐ UI/UX improvements and responsive design
- ☐ Performance optimization
- ☐ Error handling and loading states
- ☐ Image optimization and lazy loading
- ☐ PWA features
- ☐ Testing and bug fixes

Phase 6: Advanced Features (Week 11-12)

Priority: Low (Nice to have)

- ☐ Travel analytics dashboard
- ☐ Social features (likes, comments)
- ☐ Trip collaboration features
- ☐ Export to PDF functionality
- ☐ Mobile app optimization
- ☐ Advanced search filters

MVP (Minimum Viable Product) Features

Must-Have Features for Hackathon

1. **User Authentication** - Login, register, profile

2. **Trip Management** - Create, edit, delete trips
3. **Itinerary Builder** - Add destinations and activities
4. **Budget Tracking** - Basic budget allocation and tracking
5. **Search Cities** - Integration with external city API
6. **Trip Sharing** - Generate shareable public links
7. **Responsive Design** - Works on mobile and desktop

Nice-to-Have Features

1. **Weather Integration** - Weather info for destinations
2. **Currency Conversion** - Multi-currency support
3. **Photo Integration** - Destination photos from Unsplash
4. **Advanced Analytics** - Travel statistics and insights
5. **Collaboration** - Multiple users editing same trip
6. **Export Options** - PDF export of itineraries

🔧 Development Setup Guide

Prerequisites

- Node.js (v16+)
- MongoDB Atlas account
- Cloudinary account
- External API keys (Weather, Unsplash, etc.)

Quick Start

Clone repository

```
git clone <repository-url>
```

```
cd globetrotter
```

Backend setup

```
cd backend
```

```
npm install
```

```
cp .env.example .env
```

Configure environment variables

```
npm run dev
```

Frontend setup

cd ../frontend

npm install

cp .env.example .env.local

Configure environment variables

npm start

Environment Variables

Backend (.env)

NODE_ENV=development

PORT=5000

MONGODB_URI=mongodb+srv://username:password@cluster.mongodb.net/globetrotter

JWT_SECRET=your-super-secret-jwt-key

JWT_REFRESH_SECRET=your-refresh-secret-key

JWT_EXPIRE=7d

JWT_REFRESH_EXPIRE=30d

Cloudinary Configuration

CLOUDINARY_CLOUD_NAME=your-cloud-name

CLOUDINARY_API_KEY=your-api-key

CLOUDINARY_API_SECRET=your-api-secret

Email Configuration

EMAIL_FROM=noreply@globetrotter.com

SENDGRID_API_KEY=your-sendgrid-key

OR for Gmail

EMAIL_USER=your-email@gmail.com

EMAIL_PASS=your-app-password

External APIs

WEATHER_API_KEY=your-openweathermap-key

UNSPLASH_ACCESS_KEY=your-unsplash-key

```
EXCHANGE_RATE_API_KEY=your-exchange-rate-key  
GOOGLE_CLIENT_ID=your-google-oauth-client-id  
GOOGLE_CLIENT_SECRET=your-google-oauth-secret
```

```
# CORS
```

```
CLIENT_URL=http://localhost:3000
```

Frontend (.env.local)

```
REACT_APP_API_URL=http://localhost:5000/api  
REACT_APP_GOOGLE_CLIENT_ID=your-google-oauth-client-id  
REACT_APP_CLOUDINARY_CLOUD_NAME=your-cloud-name  
REACT_APP_UNSPASH_ACCESS_KEY=your-unsplash-key
```

Mobile Optimization

Responsive Design Breakpoints

```
/* Mobile First Approach */  
/* Mobile: 320px - 768px */  
/* Tablet: 768px - 1024px */  
/* Desktop: 1024px+ */
```

```
@media (max-width: 768px) {  
  /* Mobile specific styles */  
}
```

```
@media (min-width: 769px) and (max-width: 1024px) {  
  /* Tablet specific styles */  
}
```

```
@media (min-width: 1025px) {  
  /* Desktop specific styles */  
}
```

Mobile-Specific Features

- Touch-friendly interface elements

- Swipe gestures for itinerary navigation
- Offline data caching with service workers
- Mobile-optimized image loading
- GPS integration for location-based features
- Touch-optimized drag and drop

UI/UX Design Guidelines

Color Palette

```
:root {  
  
  /* Primary Colors */  
  
  --primary-blue: #3B82F6;  
  --primary-dark: #1E40AF;  
  --primary-light: #DBEAFE;  
  
  
  /* Secondary Colors */  
  
  --secondary-green: #10B981;  
  --secondary-orange: #F59E0B;  
  --secondary-purple: #8B5CF6;  
  
  
  /* Neutral Colors */  
  
  --gray-900: #111827;  
  --gray-700: #374151;  
  --gray-500: #6B7280;  
  --gray-300: #D1D5DB;  
  --gray-100: #F3F4F6;  
  --white: #FFFFFF;  
  
  
  /* Status Colors */  
  
  --success: #10B981;  
  --warning: #F59E0B;  
  --error: #EF4444;  
  --info: #3B82F6;
```

```
}
```

Typography

```
/* Font Stack */
```

```
font-family: 'Inter', -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, sans-serif;
```

```
/* Heading Styles */
```

```
.h1 { font-size: 2.5rem; font-weight: 700; }
```

```
.h2 { font-size: 2rem; font-weight: 600; }
```

```
.h3 { font-size: 1.5rem; font-weight: 600; }
```

```
.h4 { font-size: 1.25rem; font-weight: 500; }
```

```
/* Body Text */
```

```
.body-large { font-size: 1.125rem; line-height: 1.6; }
```

```
.body { font-size: 1rem; line-height: 1.5; }
```

```
.body-small { font-size: 0.875rem; line-height: 1.4; }
```

Component Design Patterns

- **Card-based layouts** for trip and destination displays
- **Timeline components** for itinerary visualization
- **Modal dialogs** for forms and confirmations
- **Toast notifications** for user feedback
- **Loading skeletons** for better perceived performance
- **Empty states** with helpful CTAs

Data Flow Architecture

State Management Strategy

```
// Global State (Context + useReducer)
```

```
const AppContext = {  
  user: { profile, settings, stats },  
  trips: { list, current, filters },  
  ui: { loading, errors, modals },  
  cache: { cities, activities, weather }  
};
```

// Component Level State (useState)

- Form data
- Component-specific UI state
- Temporary selections
- Local loading states

API Data Flow

Frontend Component →

API Service →

Express Route →

Controller →

Database/External API →

Response Back Through Chain



Deployment Checklist

Pre-Deployment

- ☐ All environment variables configured
- ☐ Database indexes created
- ☐ API rate limiting implemented
- ☐ CORS properly configured
- ☐ Error handling tested
- ☐ Performance optimization completed
- ☐ Security headers configured
- ☐ SSL certificates ready

Deployment Steps

- ☐ Build production bundle
- ☐ Deploy backend to Railway/Render
- ☐ Deploy frontend to Vercel/Netlify
- ☐ Configure custom domain (if available)
- ☐ Set up monitoring and logging
- ☐ Test all features in production
- ☐ Set up backup strategy

Post-Deployment

- [] Monitor error rates and performance
- [] Set up analytics tracking
- [] Configure automated backups
- [] Set up uptime monitoring
- [] Document API endpoints
- [] Create user guide/documentation



Success Metrics

Technical Metrics

- **Performance:** Page load time < 3 seconds
- **Reliability:** 99.5% uptime
- **Security:** Zero security vulnerabilities
- **Mobile:** Responsive on all screen sizes
- **API:** Average response time < 500ms

User Experience Metrics

- **Usability:** Users can create a trip in < 5 minutes
- **Engagement:** Average session duration > 10 minutes
- **Retention:** 70% of users return within a week
- **Completion:** 80% of started trips are completed
- **Sharing:** 30% of trips are shared publicly

Feature Adoption

- **Trip Creation:** 95% of registered users create at least one trip
- **Itinerary Builder:** 80% add multiple destinations
- **Budget Tracking:** 60% use budget features
- **Sharing:** 40% share at least one trip
- **Search:** 70% use city/activity search

This comprehensive roadmap provides a complete foundation for building GlobeTrotter as a fully-featured travel planning application using the MERN stack. The structure is designed to be scalable, maintainable, and user-friendly, with clear development phases and success metrics to guide the implementation process.