Traveler Connect & Request - Web App Specifications

Project Overview

Project Name: Traveler Connect & Request
Purpose: A platform connecting travelers with people who need items brought from other cities/countries
Target Audience: International and domestic travelers, people seeking items from specific locations

Core Concept

Users traveling between cities/countries can post their trip details along with a service fee. Other users can search for these trips and request travelers to bring specific items (e.g., electronics, local products, etc.). Travelers include their LinkedIn profiles to establish credibility and trust.

Key Features

1. User Management System

User Registration & Authentication

- **Registration Options:**
  - Email/password combination
  - LinkedIn OAuth integration (preferred for credibility)
  - Google/Facebook social login (optional)

User Profiles

- **Required Fields:**
  - Full name
  - Email address
  - LinkedIn profile URL (mandatory for travelers)
  - Phone number (optional but recommended)
- **Optional Fields:**
  - Profile picture
  - Bio/description
  - Languages spoken

- Rating/reputation score

Profile Verification

- LinkedIn profile verification
- Email verification
- Optional phone verification

2. Trip Posting System

Trip Creation Form

- **Origin Location:** City/Country with autocomplete suggestions
- **Destination Location:** City/Country with autocomplete suggestions
- **Travel Date:** Date picker with time selection
- **Return Date:** Optional return journey date
- **Service Fee:** Amount in local currency (INR, USD, etc.)
- **LinkedIn Profile:** Direct link to traveler's LinkedIn
- **Additional Notes:** Free text for special conditions/limitations
- **Baggage Capacity:** Optional field for space/weight limitations

Trip Management

- Edit existing trips
- Cancel trips
- Mark trips as completed
- View trip analytics (views, requests received)

3. Search & Discovery System

Search Functionality

- **Primary Filters:**
  - From location (city/country)
  - To location (city/country)
  - Date range selection
  - Maximum service fee
- **Advanced Filters:**
  - Traveler rating
  - Baggage capacity

- Return journey availability
- Language preferences

Trip Listings Display

- **Trip Card Information:**
  - Route: Origin → Destination
  - Travel date and time
  - Service fee amount
  - Traveler's name and LinkedIn link
  - Traveler rating/reviews
  - "Request Item" button

4. Item Request System

Request Creation

- **Request Form Fields:**
  - Item description (detailed)
  - Estimated size/weight
  - Urgency level
  - Maximum budget for item + service fee
  - Special handling instructions
  - Delivery location preference

Request Management Workflow

1. **User submits request → Traveler receives notification**
2. **Traveler reviews request → Accept/Decline/Counter-offer**
3. **If accepted → Both parties coordinate details**
4. **Item purchased/collected → Status updated to "In Transit"**
5. **Item delivered → Status updated to "Completed"**
6. **Feedback and rating exchange**

5. Communication System

In-App Messaging

- Direct messaging between travelers and requesters
- Message history for each request
- File sharing capability (receipts, photos)
- Read receipts and online status

Notification System

- Email notifications for new requests
- SMS notifications (optional)
- Push notifications for mobile app
- In-app notification center

6. Trust & Safety Features

LinkedIn Integration

- Mandatory LinkedIn profile for travelers
- LinkedIn profile verification
- Display LinkedIn endorsements/connections
- Option to message via LinkedIn

Rating & Review System

- 5-star rating system
- Written reviews and feedback
- Mutual rating (traveler rates requester, vice versa)
- Review moderation system

Safety Measures

- Report inappropriate behavior
- Block/unblock users
- Transaction dispute resolution
- Terms of service and community guidelines

7. Payment & Pricing System

Service Fee Structure

- Travelers set their own service fees
- Fee displayed in local currency
- Optional: Platform commission (future feature)
- Multiple currency support

Payment Methods (Future Implementation)

- Currently: External payment coordination
- Future: Integrated payment gateway
- Escrow service for high-value items
- Payment protection and refunds

Technical Architecture

Frontend Requirements

Technology Stack

- **Framework:** React.js with Next.js for SSR/SEO
- **Styling:** Tailwind CSS or Material-UI
- **State Management:** Redux Toolkit or Context API
- **Form Handling:** React Hook Form with validation
- **Maps Integration:** Google Maps API for location services

Key Pages Structure

```
/
├── Home (Search & Browse Trips)
├── /post-trip (Create New Trip)
├── /trips (My Trips Dashboard)
├── /requests (My Requests Dashboard)
├── /profile (User Profile & Settings)
├── /trip/[id] (Trip Details)
├── /user/[id] (Public User Profile)
├── /messages (Communication Center)
└── /auth (Login/Register)
```

Backend Requirements

Technology Stack

- **Runtime:** Node.js with Express.js or Python with FastAPI
- **Database:** PostgreSQL with Prisma ORM
- **Authentication:** NextAuth.js or Auth0
- **File Storage:** AWS S3 or Cloudinary
- **Email Service:** SendGrid or AWS SES
- **Real-time:** Socket.io for messaging

Database Schema

```
-- Users Table
CREATE TABLE users (
    id SERIAL PRIMARY KEY,
    email VARCHAR(255) UNIQUE NOT NULL,
    name VARCHAR(255) NOT NULL,
    linkedin_url VARCHAR(500) NOT NULL,
    phone VARCHAR(20),
    profile_picture VARCHAR(500),
    bio TEXT,
    rating DECIMAL(2,1) DEFAULT 0,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Trips Table
CREATE TABLE trips (
    id SERIAL PRIMARY KEY,
    user_id INTEGER REFERENCES users(id),
    from_city VARCHAR(255) NOT NULL,
    from_country VARCHAR(255) NOT NULL,
    to_city VARCHAR(255) NOT NULL,
    to_country VARCHAR(255) NOT NULL,
    travel_date TIMESTAMP NOT NULL,
    return_date TIMESTAMP,
    service_fee DECIMAL(10,2) NOT NULL,
    currency VARCHAR(3) NOT NULL,
    notes TEXT,
    baggage_capacity VARCHAR(255),
    status VARCHAR(50) DEFAULT 'active',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Requests Table
CREATE TABLE requests (
    id SERIAL PRIMARY KEY,
    trip_id INTEGER REFERENCES trips(id),
    requester_id INTEGER REFERENCES users(id),
    item_description TEXT NOT NULL,
```

```
    estimated_size VARCHAR(255),
    urgency_level VARCHAR(50),
    max_budget DECIMAL(10,2),
    delivery_location TEXT,
    status VARCHAR(50) DEFAULT 'pending',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Messages Table
CREATE TABLE messages (
    id SERIAL PRIMARY KEY,
    request_id INTEGER REFERENCES requests(id),
    sender_id INTEGER REFERENCES users(id),
    receiver_id INTEGER REFERENCES users(id),
    message TEXT NOT NULL,
    file_url VARCHAR(500),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Reviews Table
CREATE TABLE reviews (
    id SERIAL PRIMARY KEY,
    request_id INTEGER REFERENCES requests(id),
    reviewer_id INTEGER REFERENCES users(id),
    reviewee_id INTEGER REFERENCES users(id),
    rating INTEGER CHECK (rating >= 1 AND rating <= 5),
    comment TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

API Endpoints

Authentication

- `POST /api/auth/register` - User registration
- `POST /api/auth/login` - User login
- `POST /api/auth/logout` - User logout
- `GET /api/auth/me` - Get current user

Trips

- `GET /api/trips` - Search and filter trips
- `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

Requests

- `POST /api/requests` - Create item request
- `GET /api/requests` - Get user's requests
- `PUT /api/requests/:id` - Update request status
- `DELETE /api/requests/:id` - Cancel request

Users

- `GET /api/users/:id` - Get user profile
- `PUT /api/users/:id` - Update user profile
- `GET /api/users/:id/reviews` - Get user reviews

Messages

- `GET /api/messages/:requestId` - Get conversation
- `POST /api/messages` - Send message

User Experience Flow

For Travelers

1. **Account Creation**
    - Register with LinkedIn authentication
    - Complete profile with verified LinkedIn URL
    - Set up notification preferences
2. **Trip Posting**
    - Navigate to "Post a Trip" page
    - Fill out travel details and service fee
    - Publish trip for public viewing
3. **Request Management**
    - Receive notifications for new requests
    - Review request details and requester profile
    - Accept, decline, or negotiate requests
    - Coordinate purchase and delivery details
4. **Completion & Feedback**
    - Update request status throughout journey
    - Receive payment for services
    - Leave feedback for requesters

For Requesters

1. **Trip Discovery**
   - Search trips by origin/destination
   - Filter by date, price, and traveler rating
   - View traveler LinkedIn profiles for credibility
2. **Request Submission**
   - Select suitable trip and traveler
   - Submit detailed item request
   - Await traveler response
3. **Coordination & Communication**
   - Communicate via in-app messaging
   - Provide item details and payment
   - Track request status
4. **Completion & Review**
   - Receive requested item
   - Complete payment to traveler
   - Leave feedback and rating

Security & Privacy

Data Protection

- GDPR compliance for EU users
- Encrypted data transmission (HTTPS)
- Secure password hashing
- Personal data anonymization options

Platform Safety

- LinkedIn verification requirement
- User reporting and blocking system
- Content moderation for inappropriate requests
- Community guidelines enforcement

Financial Security

- No storage of payment card details
- Secure payment processing (when implemented)
- Transaction history and receipts
- Fraud detection and prevention

Deployment & Infrastructure

Hosting Requirements

- **Frontend:** Vercel, Netlify, or AWS Amplify
- **Backend:** AWS EC2, Google Cloud, or Heroku
- **Database:** AWS RDS, Google Cloud SQL, or Supabase
- **CDN:** CloudFlare for static assets
- **Monitoring:** DataDog, New Relic, or Sentry

Environment Configuration

```
# Database
DATABASE_URL=postgresql://username:password@host:port/database

# Authentication
NEXTAUTH_SECRET=your-secret-key
LINKEDIN_CLIENT_ID=your-linkedin-client-id
LINKEDIN_CLIENT_SECRET=your-linkedin-client-secret

# External Services
GOOGLE_MAPS_API_KEY=your-google-maps-key
SENDGRID_API_KEY=your-sendgrid-key
AWS_ACCESS_KEY_ID=your-aws-key
AWS_SECRET_ACCESS_KEY=your-aws-secret
```

Development Roadmap

Phase 1: MVP (Minimum Viable Product)

- [x] User registration and authentication
- [x] Basic trip posting and searching
- [x] Simple request system
- [x] LinkedIn profile integration
- [x] Basic messaging system

Phase 2: Enhanced Features

- [ ] Advanced search filters
- [ ] Rating and review system
- [ ] Email notifications
- [ ] Mobile responsive design
- [ ] File sharing in messages

Phase 3: Advanced Platform

- [ ] Mobile app development
- [ ] Integrated payment system
- [ ] Multi-language support
- [ ] Advanced analytics dashboard
- [ ] API for third-party integrations

Phase 4: Scale & Optimize

- [ ] Machine learning recommendations
- [ ] Automated matching system
- [ ] Business traveler features
- [ ] Corporate partnerships
- [ ] Revenue optimization

Success Metrics

User Engagement

- Monthly Active Users (MAU)
- Trip posting frequency
- Request completion rate
- User retention rate

Platform Growth

- New user registrations
- Geographic expansion
- Trip volume growth
- Revenue per transaction

Quality Metrics

- Average user rating
- Dispute resolution time
- LinkedIn verification rate
- Platform safety incidents

Legal & Compliance

Terms of Service

- Platform liability limitations
- User responsibility definitions
- Dispute resolution procedures
- Account termination policies

Privacy Policy

- Data collection and usage
- Third-party integrations
- Cookie policy
- User rights and controls

International Considerations

- Customs and import regulations
- International shipping restrictions
- Currency exchange regulations
- Cross-border payment compliance

GitHub Copilot Development Instructions

When implementing this specification:

1. **Start with the MVP features** listed in Phase 1
2. **Prioritize user authentication** and LinkedIn integration first
3. **Implement core trip posting and searching** functionality
4. **Build the request system** with basic status management
5. **Add messaging capabilities** for user communication
6. **Focus on responsive design** and user experience
7. **Implement proper error handling** and validation

8. **Add comprehensive testing** for all features
9. **Ensure security best practices** throughout development
10. **Document all API endpoints** and component usage

Use this specification as your complete reference for building the Traveler Connect & Request web application. Each section provides detailed requirements that should guide your development decisions and implementation approach.