🚀 DEVELOPER CHEAT SHEET

Enterprise CRM - Quick Reference

🔧 LOCAL DEVELOPMENT SETUP

1. Start Backend (Django)

```bash
cd backend
source venv/bin/activate  # or venv\Scripts\activate on Windows
python manage.py runserver
# Runs on http://localhost:8000
```

2. Start Frontend (React)

```bash
cd frontend
npm start
# Runs on http://localhost:3000
```

3. Database Migrations

```bash
python manage.py makemigrations
python manage.py migrate
```

4. Create Superuser

```bash
python manage.py createsuperuser
```

📝 TESTING APIs WITH CURL

Register User

```bash
curl -X POST http://localhost:8000/api/v1/auth/register/ \
  -H "Content-Type: application/json" \
  -d '{
    "email": "user@example.com",
    "password": "SecurePass123!",
    "password_confirm": "SecurePass123!",
    "first_name": "John",
    "last_name": "Doe"
  }'
```

Login

```bash
curl -X POST http://localhost:8000/api/v1/auth/login/ \
  -H "Content-Type: application/json" \
  -d '{
    "email": "user@example.com",
    "password": "SecurePass123!"
  }'
# Save the access_token from response
```

Create Account (with token)

```bash
curl -X POST http://localhost:8000/api/v1/accounts/ \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer YOUR_ACCESS_TOKEN" \
  -d '{
    "name": "Acme Corp",
    "email": "contact@acme.com",
    "account_type": "customer",
    "industry": "Technology"
  }'
```

List Accounts

```bash
curl -X GET http://localhost:8000/api/v1/accounts/ \
  -H "Authorization: Bearer YOUR_ACCESS_TOKEN"
```

Search Accounts

```bash
curl -X GET "http://localhost:8000/api/v1/accounts/?search=acme" \
  -H "Authorization: Bearer YOUR_ACCESS_TOKEN"
```

🗂️ FILE LOCATIONS

Backend

```
backend/
├── core/
│   ├── models.py              # User, Company models
│   ├── serializers/auth.py    # Auth serializers
│   ├── views/auth.py          # Auth endpoints
│   ├── middleware.py          # Multi-tenant middleware
│   └── urls.py                # Auth URLs
│
├── crm/
│   ├── models.py              # Account model
│   ├── models/contacts.py     # Contact model
│   ├── serializers/
│   │   ├── accounts.py        # Account serializers
│   │   └── contacts.py        # Contact serializers
│   └── views/
│       ├── accounts.py        # Account ViewSet
│       └── contacts.py        # Contact ViewSet
│
└── config/
    ├── settings/base.py       # Django settings
    └── urls.py                # Root URLs
```

Frontend

```
frontend/src/
├── api/
│   ├── client.js            # Axios base client
│   ├── accounts.js          # Account API
│   └── contacts.js          # Contact API
│
├── pages/
│   ├── Accounts/
│   │   └── AccountsList.jsx   # Accounts list page
│   └── Contacts/
│       └── ContactsList.jsx   # Contacts list page
│
├── context/
│   ├── AuthContext.jsx        # Authentication context
│   └── CompanyContext.jsx     # Company context
│
└── hooks/
    └── useAuth.js           # Auth hook
```

🔑 KEY CONCEPTS

Multi-Tenancy

Every model that needs company isolation extends `CompanyIsolatedModel`:

```python
class Account(CompanyIsolatedModel):
    name = models.CharField(max_length=255)
    # company field is automatically added
    # created_by, updated_by, created_at, updated_at also added
```

ViewSet Pattern

```python
```

```
class AccountViewSet(viewsets.ModelViewSet):
    serializer_class = AccountSerializer
    permission_classes = [IsAuthenticated]

    def get_queryset(self):
        # Always filter by active company
        return Account.objects.filter(
            company=self.request.active_company
        )
```

React API Call Pattern

javascript

```
// In component
const [data, setData] = useState([]);

useEffect(() => {
  const fetchData = async () => {
    const response = await accountsAPI.getAll();
    setData(response.data.results);
  };
  fetchData();
}, []);
```

📊 DATABASE QUERIES

Get all accounts for a company

sql

```
SELECT * FROM accounts WHERE company_id = 'uuid-here';
```

Get user's companies

sql

```
SELECT c.* FROM companies c
JOIN user_company_access uca ON c.id = uca.company_id
WHERE uca.user_id = 'user-uuid' AND uca.is_active = true;
```

Get contacts with accounts

```sql
SELECT c.*, a.name as account_name
FROM contacts c
LEFT JOIN accounts a ON c.account_id = a.id
WHERE c.company_id = 'uuid-here';
```

🎨 COMMON UI PATTERNS

Loading State

```jsx
{loading ? (
  <div className="animate-spin rounded-full h-12 w-12 border-b-2 border-blue-600"></div>
) : (
  // Content
)}
```

Empty State

```jsx
{items.length === 0 ? (
  <div className="text-center py-12">
    <Icon className="mx-auto h-12 w-12 text-gray-400" />
    <h3 className="mt-2 text-sm font-medium">No items</h3>
  </div>
) : (
  // List
)}
```

Search Input

```jsx
```

```
<div className="relative">
  <Search className="absolute left-3 top-1/2 transform -translate-y-1/2" />
  <input
    type="text"
    placeholder="Search..."
    className="w-full pl-10 pr-4 py-2 border rounded-lg"
  />
</div>
```

🔍 DEBUGGING

Check Active Company

```python
# In Django view
print(request.active_company)
print(request.session.get('active_company_id'))
```

Check User Permissions

```python
# In Django view
print(request.permissions)
```

Check API Response

```javascript
// In React
console.log('Response:', response.data);
```

Check Database Connection

```bash
python manage.py dbshell
\dt  # List all tables
```

🛠️ COMMON TASKS

Add New Field to Model

1. Add field to model:

```python
new_field = models.CharField(max_length=100, blank=True)
```

2. Make migration:

```bash
python manage.py makemigrations
```

3. Apply migration:

```bash
python manage.py migrate
```

4. Add to serializer fields list
5. Update frontend if needed

Add New API Endpoint

1. Add method to ViewSet:

```python
@action(detail=True, methods=['get'])
def custom_action(self, request, pk=None):
    return Response({'message': 'Success'})
```

2. Add to API client:

```javascript
customAction: (id) => apiClient.get(`/accounts/${id}/custom-action/`)
```

Add New Filter

1. Add to `filterset_fields`:

```python
filterset_fields = ['field1', 'field2', 'new_field']
```

2. Add filter UI in React

3. Pass in params to API call

📦 PRODUCTION DEPLOYMENT

Environment Variables (.env)

```env
DEBUG=False
SECRET_KEY=your-secret-key-here
DB_NAME=crm_production
DB_USER=crm_user
DB_PASSWORD=secure-password
DB_HOST=db.example.com
DB_PORT=5432
ALLOWED_HOSTS=api.yoursite.com
CORS_ALLOWED_ORIGINS=https://app.yoursite.com
```

Collect Static Files

```bash
python manage.py collectstatic --noinput
```

Build Frontend

```bash
cd frontend
npm run build
# Outputs to build/ directory
```

🐛 COMMON ERRORS & FIXES

"company_id cannot be null"

Fix: Ensure middleware is setting `request.active_company`

"No module named 'core'"

Fix: Add app to INSTALLED_APPS in settings.py

CORS Error in Frontend

Fix: Add frontend URL to CORS_ALLOWED_ORIGINS

"relation does not exist"

Fix: Run migrations: `python manage.py migrate`

401 Unauthorized

Fix: Check JWT token is being sent in Authorization header

📚 USEFUL COMMANDS

Django

```bash
# Create new app
python manage.py startapp app_name

# Open Django shell
python manage.py shell

# Check for issues
python manage.py check

# Show migrations
python manage.py showmigrations

# Create admin user
python manage.py createsuperuser

# Run tests
python manage.py test
```

React

```bash
# Install package
npm install package-name

# Remove package
npm uninstall package-name

# Check for outdated packages
npm outdated

# Update packages
npm update
```

Git

```bash
# Commit changes
git add .
git commit -m "Add feature"
git push

# Create new branch
git checkout -b feature-name

# Merge branch
git checkout main
git merge feature-name
```

🎯 QUICK WINS

Performance

- Add `select_related()` for foreign keys
- Add `prefetch_related()` for many-to-many
- Use `only()` to limit fields returned
- Add database indexes on frequently queried fields

Security

- Always use HTTPS in production
- Set DEBUG=False
- Use strong SECRET_KEY
- Implement rate limiting
- Sanitize user inputs

User Experience

- Add loading states
- Show error messages
- Implement optimistic updates
- Add keyboard shortcuts
- Use toast notifications

📞 NEED HELP?

Resources

- Django Docs: https://docs.djangoproject.com/
- DRF Docs: https://www.django-rest-framework.org/
- React Docs: https://react.dev/
- Tailwind Docs: https://tailwindcss.com/

In This Project

- Check Implementation Guide artifact
- Review Progress Summary artifact
- Look at existing code patterns
- Test API with curl/Postman

Keep this handy while developing! 🚀