*Hassan Badru - May 2018*

# Encoder/Decoder - Art+Logic Exercise

## Description

This is a single page web application that allows users to encode and decode data

## Set Up / Running App

- Download & unpack Hassan_Badru_Part1.zip file
- (*Recommended*) Install & run a Virtual Environment (e.g. `source Art_Logic_Env/bin/activate`)
- Assuming you already have *python* and *pip*, install requirements using `pip install -r requirements.txt` within command prompt or terminal *(if not, check out How to install python & pip )*
- Assuming you already have *Node* or *NPM*, inside the **frontend** folder, install node_modules (dependencies) using `npm install` *(if not, check out How to install Node )*
- To start server, run the command `python manage.py runserver`
- On your browser, go to http://127.0.0.1:8000/ or local server address provided within terminal

*Note: On your browser, go to http://127.0.0.1:8000/api to access REST API*

## Exercise Requirements

For this task, you need to write a small program including a pair of functions that can

- Convert an integer into a special text encoding and then
- Convert the encoded value back into the original integer.

  Assuming that your solution works correctly and cleanly enough to move forward in this process, these functions will need to be used in your part 2 submission.

### The Encoding Function

My encoding function accepts a signed integer in the 14-bit range *[-8192..+8191]* and returns a 4 character hexadecimal string. The encoding process is as follows:

1. Added 8192 to the raw value, so its range is translated to *[0..16383]*
2. Packed that value into two bytes such that the most significant bit of each is cleared
3. Formatted the two bytes as a single 4-character hexadecimal string and returned it.

### The Decoding Function

My decoding function accepts two bytes on input, both in the range *[0x00..0x7F]* and recombines them to return the corresponding integer between *[-8192..+8191]*

## Folder Structure

```
Hassan_Badru_Part1
    ├── README.md
    ├── art_logic
    │   ├── __init__.py
    │   ├── settings.py
    │   ├── urls.py
```

```
    │   ├── wsgi.py
    ├── art_logic_app
    │   ├── __init__.py
    │   ├── admin.py
    │   ├── apps.py
    │   ├── fixtures
    │   │   └── art_logic_app.json
    │   ├── migrations
    │   │   ├── 0001_initial.py
    │   │   ├── 0002_auto_20180515_2039.py
    │   │   ├── 0003_auto_20180519_2226.py
    │   │   ├── __init__.py
    │   │   └── __init__.pyc
    │   ├── models.py
    │   ├── myfunctions.py
    │   ├── serializers.py
    │   ├── tests.py
    │   ├── urls.py
    │   ├── views.py
    ├── db.sqlite3
    ├── frontend
    │   ├── README.md
    │   ├── build
    │   │   ├── asset-manifest.json
    │   │   ├── favicon.ico
    │   │   ├── index.html
    │   │   ├── manifest.json
    │   │   ├── service-worker.js
    │   │   └── static
    │   │       ├── css
    │   │       │   ├── main.a203576f.css
    │   │       │   └── main.a203576f.css.map
    │   │       ├── js
    │   │       │   ├── main.c96fb44e.js
```

```
    │   │       │   └── main.c96fb44e.js.map
    │   │       └── media
    │   │           └── intro-bg.fb30f247.jpg
    │   ├──
    │   ├── package.json
    │   ├── public
    │   │   ├── favicon.ico
    │   │   ├── index.html
    │   │   └── manifest.json
    │   ├── src
    │   │   ├── App.js
    │   │   ├── App.test.js
    │   │   ├── css
    │   │   │   ├── App.css
    │   │   │   └── index.css
    │   │   ├── img
    │   │   │   └── intro-bg.jpg
    │   │   ├── index.js
    │   │   ├── logo.svg
    │   │   └── registerServiceWorker.js
    │   └── yarn.lock
    ├── requirements.txt
    ├── manage.py
    └── media
        └── ConvertedData.txt
```

## Technology Stack Used

- **HTML5 / CSS (View Template)**
- **REACT JS (Frontend)**

- **Django / Python (Backend)**
- **PostgreSQL (Database)**
- **Django REST Framework (API)**
- **Node / NPM (Production Build)**

## Structure

- ### Model

```
class UserAction(models.Model):
    operation = models.CharField(max_length=100,
default='encoding')
    input = models.CharField(max_length=100,
default='8191')
    result = models.CharField(max_length=100, default='')
```

- ### View

```
class ArtLogicAPI(generics.ListCreateAPIView):
    queryset = UserAction.objects.all()
    serializer_class = UserActionSerializer


class ArtLogicApp(TemplateView):
```

```
    template_name = 'index.html'
```

- ## Template

I set the template use to load from React's production build folder in *settings.py* `'DIRS': [ os.path.join(BASE_DIR, 'frontend/build') ],`

- ## Utility (MyFunction.py)

### Encoding Function:

```
def encoder(input_num):
    ....
    return output
```

### Decoding function:

```
def decoder(input_num):
    ....
    return output
```

- ## Routing

Django Project:

```
urlpatterns = [
  url(r'^admin/', admin.site.urls),
  url(r'^', include('art_logic_app.urls'))
] + static(settings.MEDIA_URL,
document_root=settings.MEDIA_ROOT)
```

Django App (art_logic_app):

```
    url('api/', views.ArtLogicAPI.as_view() ),
    url(r'^$', views.ArtLogicApp.as_view(),
name="art_logic" )
```

- ## API / Serialization

```
class UserActionSerializer(serializers.ModelSerializer):
    class Meta:
        model = UserAction
        fields = ('operation', 'input', 'result')
```

- ## Fixtures

Preloaded with data to be encoded/decoded and written into convertedData.txt

**File:**

`art_logic_app.json`
Preloaded data with output written into convertedData.txt

- ## Static Files

**CSS:**

- App.css
- index.css

**IMAGE:**

Background Image:

- intro-bg.jpg
- 

## Media Files (ConvertedData.txt)

I set the directory to save ConvertedData.txt in *settings.py* using:

```
MEDIA_URL = '/media/'
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
```

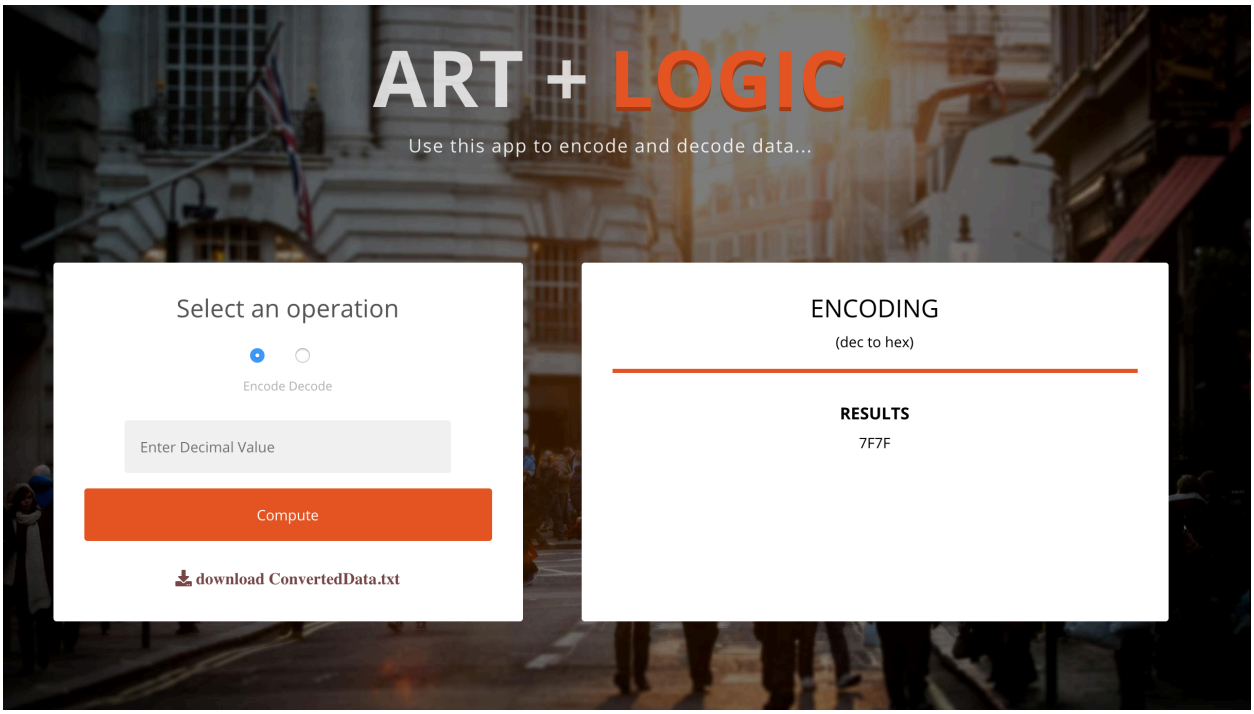
| To Encode | After Encode |
|---|---|
| 6111 | 4463 |
| 340 | 4254 |
| -2628 | 2B3C |
| -255 | 3E+01 |
| 7550 | 7A7E |

| To Decode | After Decode |
|---|---|
| 0A0A | -6902 |
| 29 | -8151 |
| 3F0F | -113 |
| 4400 | 512 |
| 5E7F | 3967 |

The app also writes all **USER ADDED** encoding/decoding conversions to the file: *ConvertedData.txt*

## How the App Works

### REACT Single Page




## Features

- User can select what kind of operation they want to perform

- After selecting operation, user can input values they want to encode/decode

- The app checks if user inputted a correctly formatted (valid) values i.e.
  14-bit signed integer (when encoding)

  16-bit hexadecimal decimal value for decoding

- Error Handling: Users get error messages if invalid values were inputted

- If no errors, the result of the encoding or decoding operation is outputted & displays.

- The app reads stored data (for encoding/decoding) in database and then uses the data object attributes to compute results

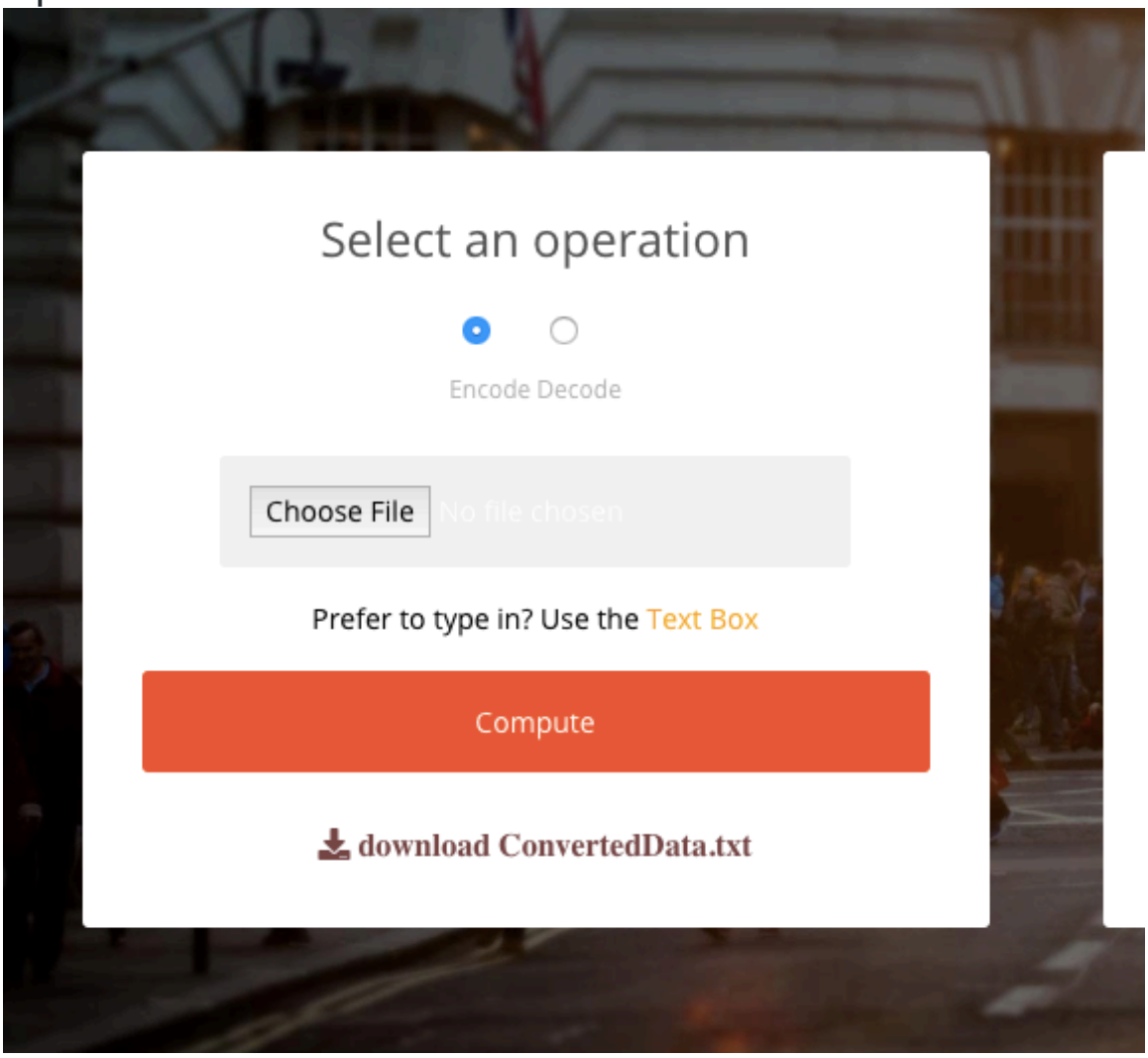
- Allows user to download convertedData.txt file containing encoding/decoding data of preloaded values

- The app keeps a record of every valid operation performed by the user and serializes it for the API

## Extensibility

- An added feature in the future could allow the userto toggle between user text input or file

input


Select an operation
Encode Decode
Choose File
No file chosen
Prefer to type in? Use the Text Box
Compute
download ConvertedData.txt