Disclaimer

1. This research document belongs to DT Group Business.co, registered in Hong Kong.
2. This document is bounded with information classification, only authorised access to personnel with Level 5 clearance or above and education institution for educational purpose.
3. This project is in research stage, it does not imply the real application in financial market, any unauthorised business activities in Hong Kong might offend the broker regulation stated in the transactions ordinance (Chapter 553).

Abstract and Objective

Hong Kong is an international city, connecting the world with financial centres and commercial ports. Hong Kong has ranked the third in Global Financial Index.

Taking the geographic advantage of Hong Kong, if business activity is going to take place in Hong Kong, investment is a good choice to begin with. As a result, a fund investment business is recommended. The objective of this research project is to find out the possibility to develop an investment business in Hong Kong. This reflective report includes the source code, a demo video and a reflective report. It is concluded that the fund project is plausible and profit sounding given the performance made in prototype framework. It is feasible to progress and present to the board of directors in

Introduction

In this project, a fund business is initiated. We aim to collect deposits across the globe and invest in certain innovation and financial products to generate greater profit. After that, the company will deduct a minimal profit percentage as handling fee and deliver back to investors through FPS.

The Structure of the Fund System

System Framework

The whole system mainly made up of these components.

1. Python coding

* Act as the interpreter of the coding
* Corporate with flask script to run the server
* Mainly support backend of the application

1. Flask

* Web framework built in python
* Redirect websites
* Collect user input and make respond
* Handle GET, POST Request
* Middleware to handle database query

1. Python module and lib used

* Flask WTForms (Taking advantage of validation function)
* SQLalchemy (ORM is more structured than relational, easier to read under declarative syntax)
* SQL Lite 3 (Cross platform available between development in Mac and Ubuntu)
* Jinja 2 template (Template function provides convenience)
* Flask-script (The script to manage the config of the development server through command line, much consistent)

1. HTML modules, plugins

* Clock (Provides user the trading time of GMT +8 HK market)
* Stock Tradingview Plugin
* Original DT Company website resources (Images, credentials and stock information)

1. CSS

* Original CSS Script CSDN given
* Bootstrap customize style design
* Buttons, body, div customize modification

1. jQuery

* Original script CDN given
* Clock effect

1. JavaScript

* Original script CDN given
* Button effect
* Clock effect
* Transition, animation Effect
* Facilitate front, back end communication, pass results between SQL

1. Bootstrap

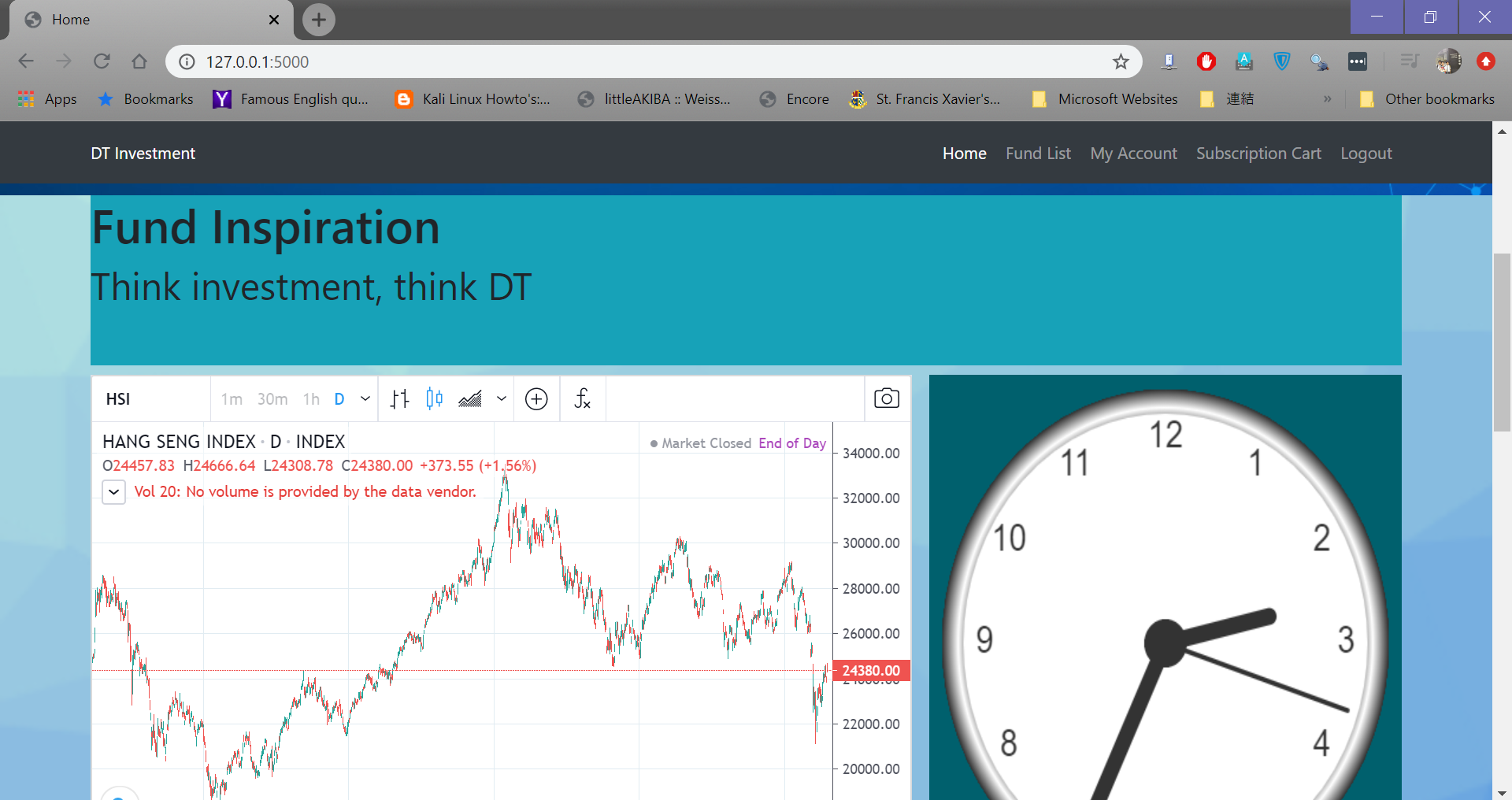
* General optimization (Providing friendly design and adopt with easy to read basis)
* Used as front end design only

1. SQL

* Access Database
* Insert, alter, modify, delete record

Front End

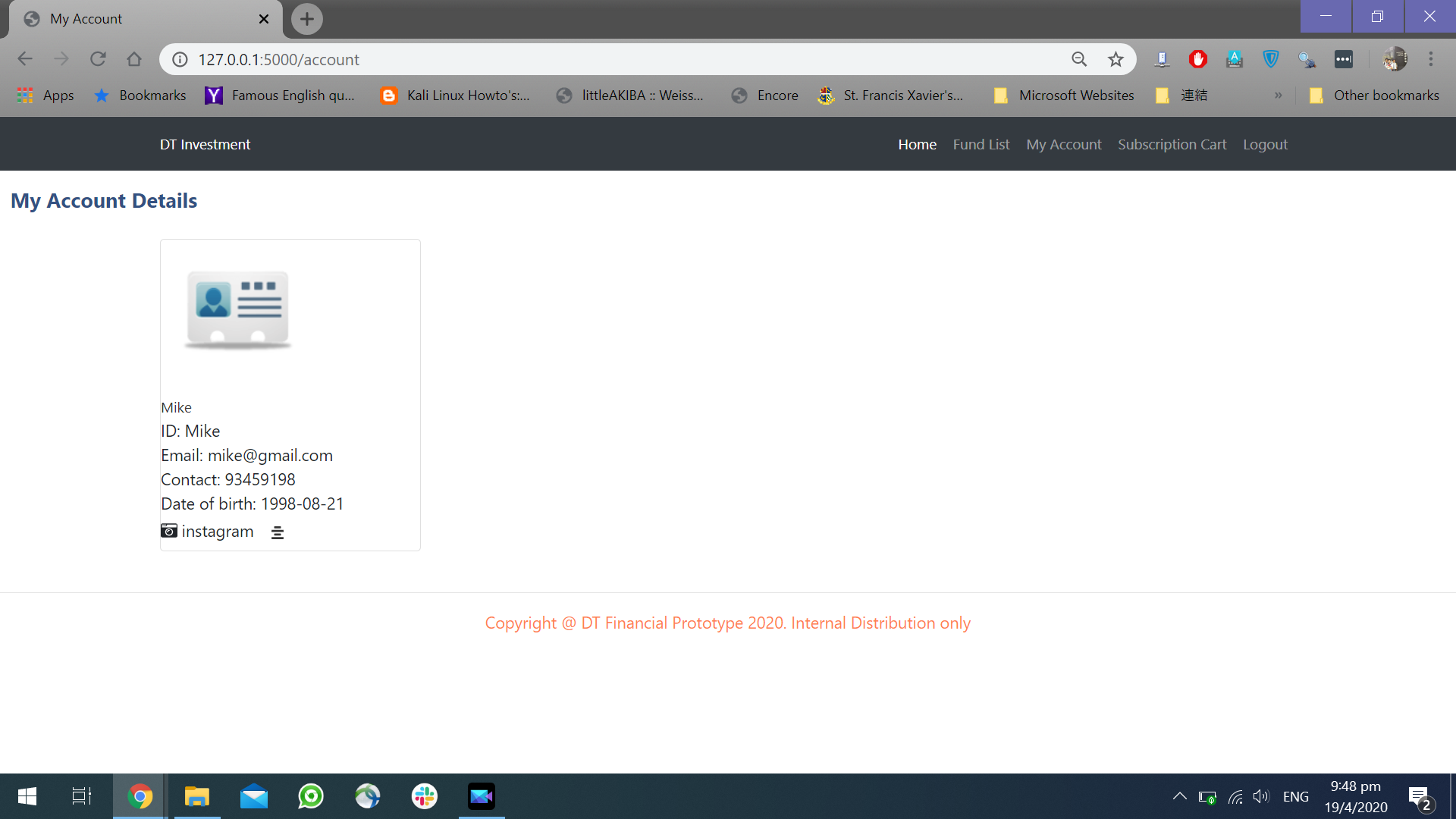
HTML design, block template techniques, CSS styling, bootstrap design technique are used to develop the front end. When the ip address is typed, it will redirect to home page automatically.



The front end gets the parameters from what is designed in backend. In the preliminary stage, all backend variables and passing in and out has been planned. This is because the backend is more complicated comparing to the visual design in frontend. The front end design is divided into module based, which helps different stuff organised one by one, easier to read and debug.

Back End

The back end is first developed before it is adopted in front end. Back end involves complicated communication between different channels and languages.



Take My Account as an example, the html needs to get customer information data from python and python gets enquiry results from database.

.py, .db, .html are the main three elements in the back end.

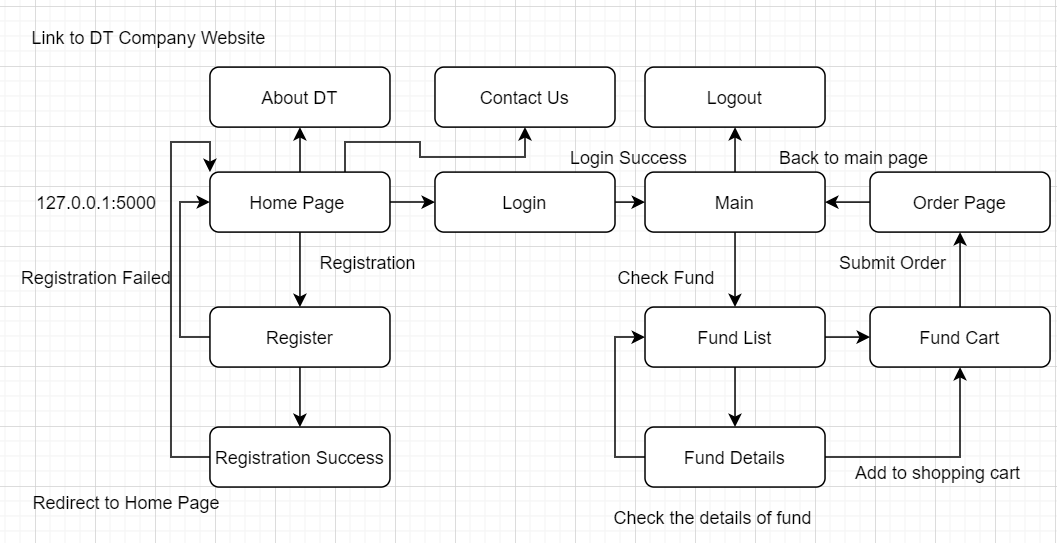
|  |  |
| --- | --- |
| .py | Logic and Algorithm are written in views.py as website routing function  The database structure are converted from .db (DBMS – SQL Lite 3) to list and dictionary format, store into computer memory and accessed through python interpreter. Form validation are also made to identify user input correctly. It is also used to invoke flask script commands through a manager module. |
| .db | The database management system to handle query from user input to python to db and db to python and display back to user through POST GET method in HTML. |
| .html | Html and javascript handle user input eg.(button on click and pass variable through base template of block template inheritance in flask). It also pass useful parameters into python flask from link “GET” for redirect purpose. |

Process description

1. Flow Chart

To let users get an overview of our business, they will be redirect to the home page if they type the following ip by default: 127.0.0.1:5000, if they need to access the system, login is a must. A session will be generated upon successful login. If they have not yet login, visitors will not hold any session that blocks them from further access. Logout eventually clear their session. A flow chart has been provided for reference.

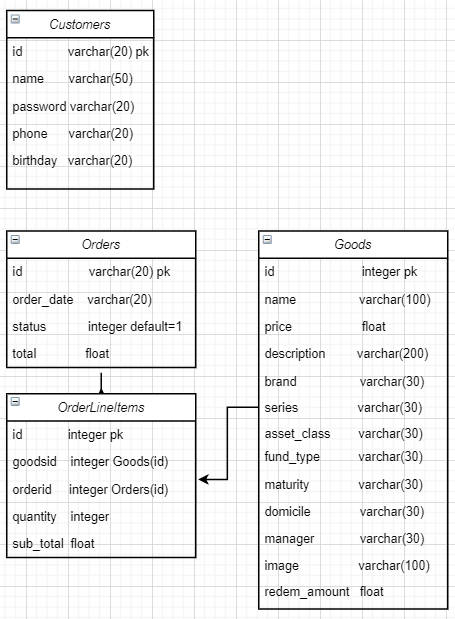
<https://app.diagrams.net/#G1Hr6YB5qpGqmoIb_VPfqT4OVqZkAOsa2Q>



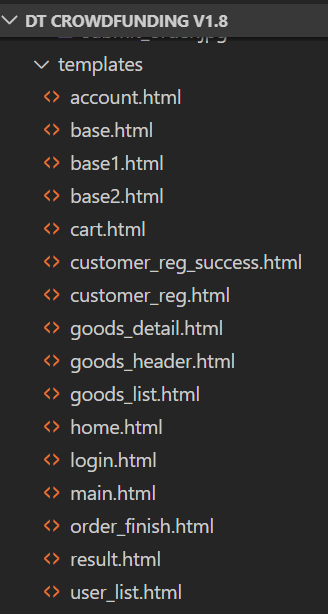
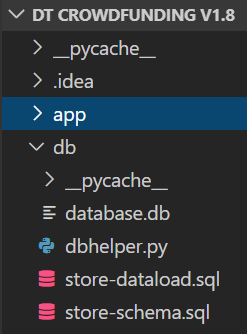
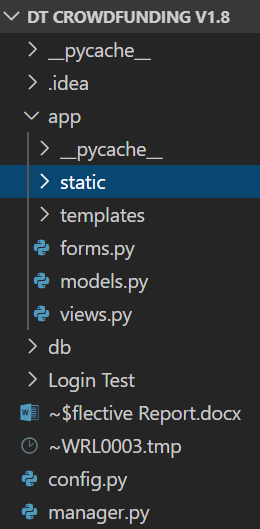
1. SQL Modal

Under ORM technique, data is converted into python readable types for access, “OrderLineItems” acts as a virtual object database that is called from python variables to link up the record in .db eg(Goods brand, series, Customer details …) object values are mapped into groups of simpler and convert them back upon retrieval. Database diagram is provided for reference.

<https://app.diagrams.net/#G1BdDUPNY1GeaSORPCGcS7JmtN_sDuE40L>



Coding Structure

The Flask application is run through manager.py. Before runserver by using the following command: “python manager.py runserver”, table is created, data must be inserted into the database from store-dataload.sql. The data organization is built this way to ensure all data are focus into one place and insert them all directly, error feedback can be easily check insertion failed and initialise failure. The database connection (dbhelper, store-schema) is also configured in a file so that it can be managed independently ignoring the web operation. All webpages are centrally managed by views.py. It operates according to the flask structure, almost each function has a session management guarding by session keys to prevent unauthorised access. Due to flexibility in data management, account page is delivered by dictionary so to suit the front-end display retrieve sequence. For each movement, flash and print function has been made use to check if errors arise. This is a kind of debugging skill to manage unexpected runtime error. The html mostly has direct relationship with views.py which will be discussed in demo video deeply.

Python Module Configuration

To configure different modules, flask script need to be installed first so that flask shell can be accessed:

|  |
| --- |
| Pip install flask-script |

Project stages and visualization

1. Collecting raw files

Images are mainly collected from the web and company website, programming environment must be set as well. And it should be compatible with different OS (Windows, Mac, Ubuntu), so to convenient different parties (Teachers, consultants, experts) giving advice and debugging. As a result, programming in python and using pip lib, sql lite are adopted. LAMP has issues in cross platform configuration that is already deprecated.

1. Programming models.py

The purpose is to identify the database data and convert into python readable format. Models receive data from .db with string, char format. Which is converted back to python variables and place it back to views.py and other python places. The approach is to create a gateway so that data can be concentratedly managed.

1. Programming forms.py

The form handles login and registration input validation which is powered by WTForm. By adopting this module, it can pre-check user’s input before it is passed to database without going through raw coding.

1. Programming views.py

Views.py is the core of the application, all websites are routed here. The design is intended to manage data flow, passing variables and data between front, backend. The system logic is also presented in here.

1. .sql Programming

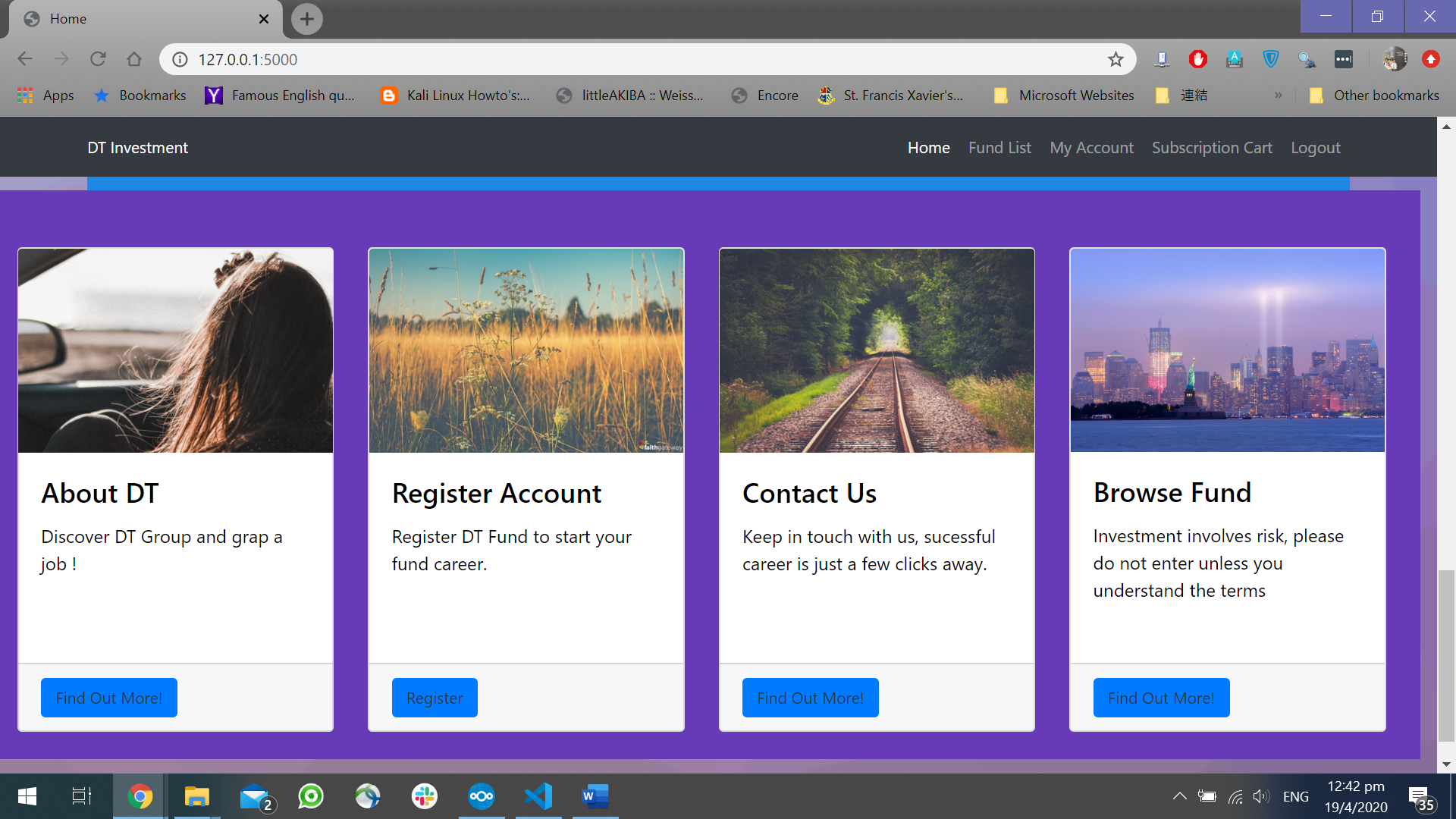
Data load acts as a USB, bunch of records pending to insert into database.

Store-schema defines the database structure, so that a table can be created without flaws. The sql has separate files because they are designed to serve manager.py for two purpose:

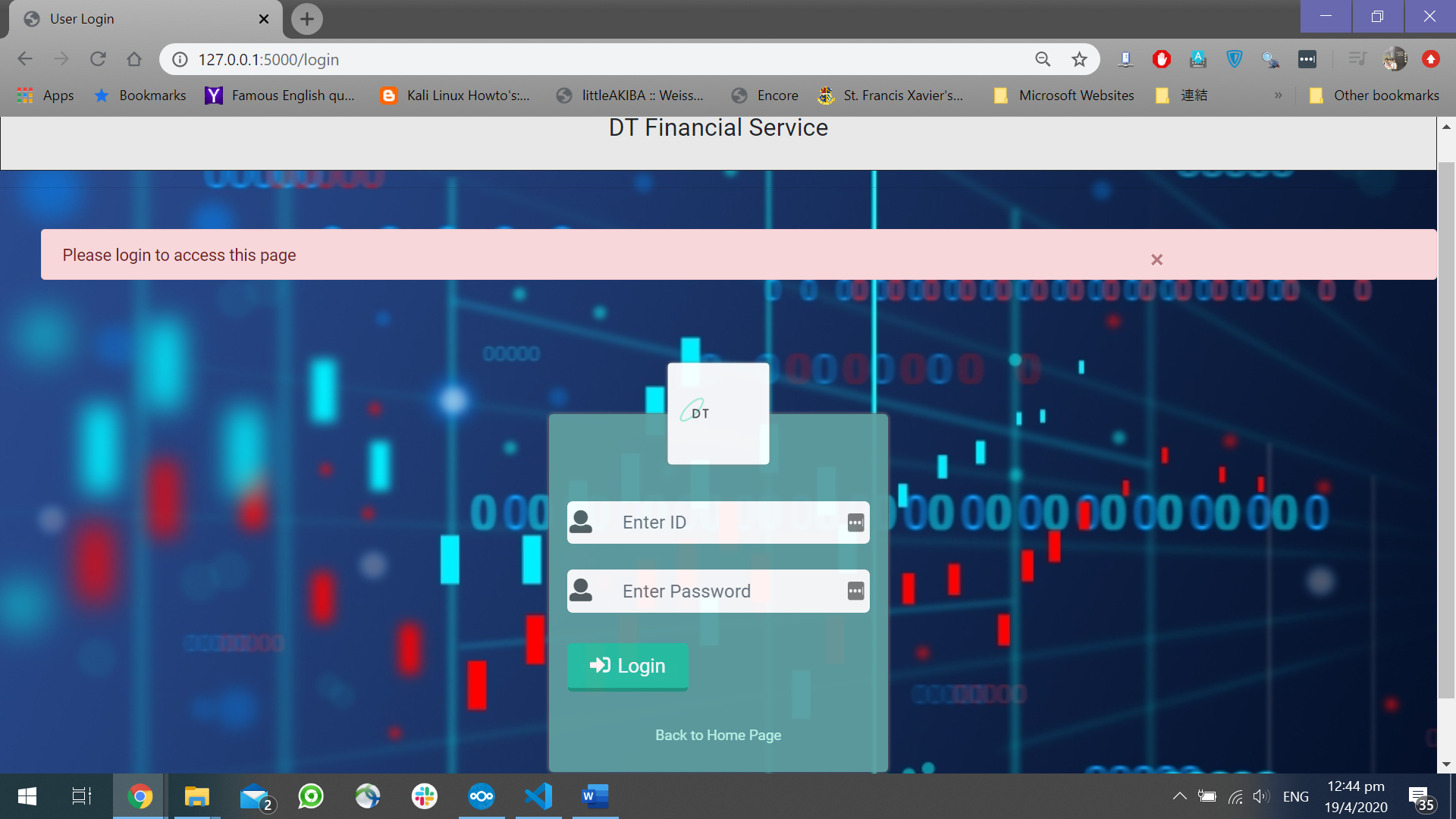
* 1. Create tables
  2. Dataload

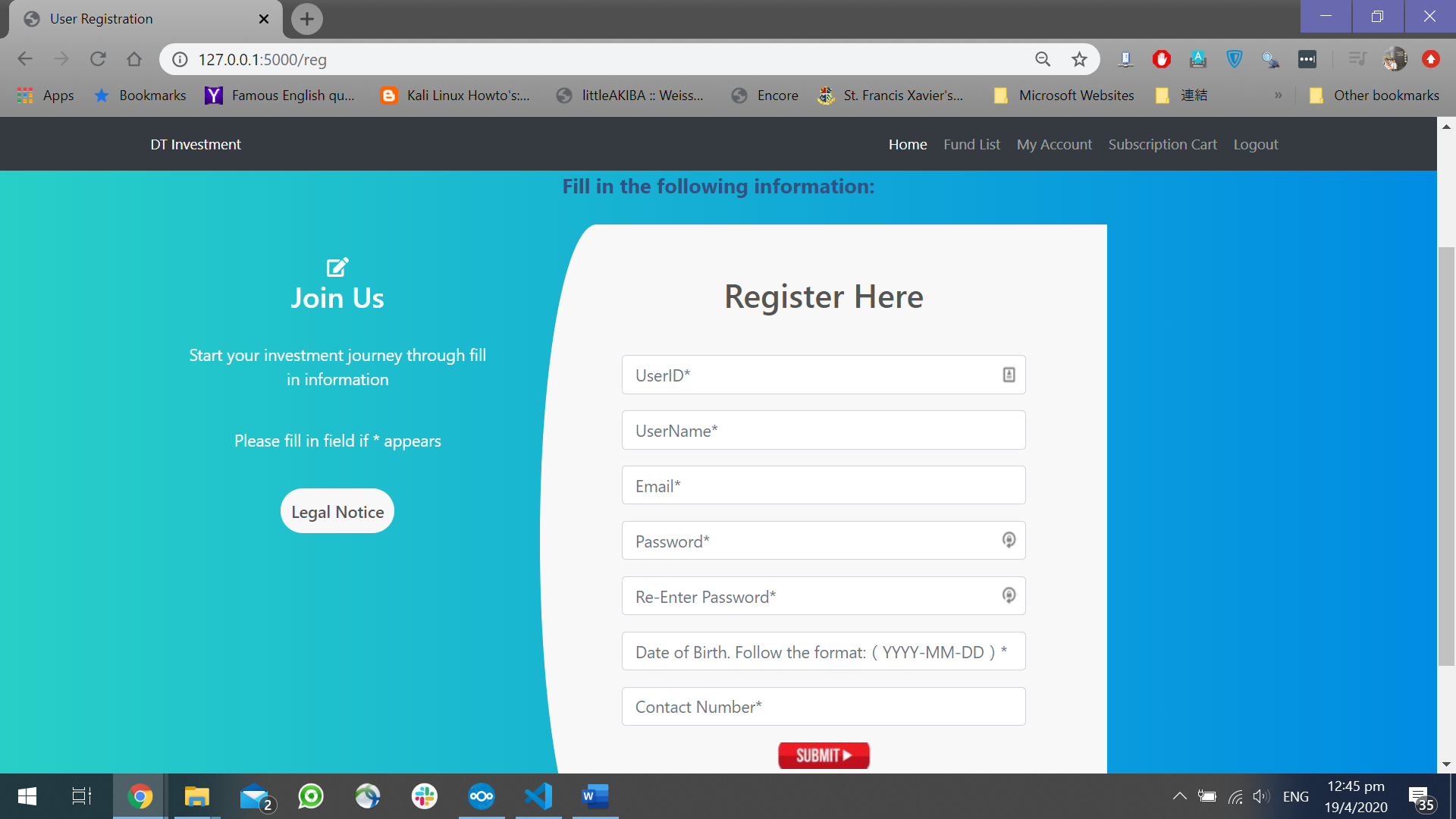
Two functions are invoked by a command via flask script, making the whole programming process easy.

Main results

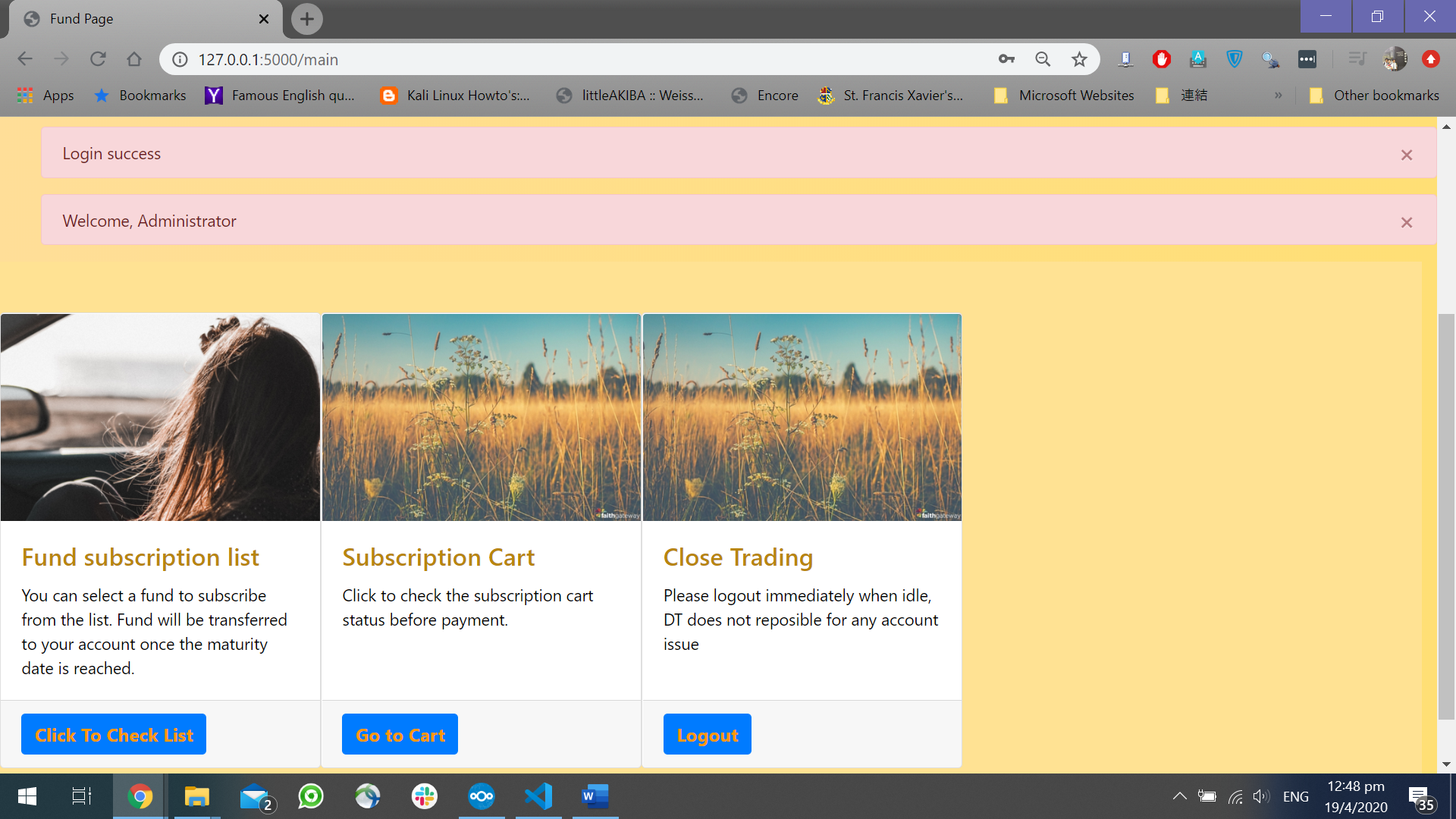


The home page acts as a navigation page, to let visitors know what our business is and give them access to all pages. This design is to attract visitors to show interest and make further navigation.





They must login or register in order to access any information about fund.



Upon successful login, they will be redirect to the trading panel. Card is used to show all function which is shown clearly to customer, so that they deploy their investment strategy with a few clicks.

Difficulties

1. It is really difficult to code the backend, although error is shown, it is hard to make adjustment as the error message is not clear to point out the error part. Unlike python which debugging mode is available for step in and out, in web app, only print, flash, javascript alert is the best tool to show step by step process.
2. Time is not enough. It is useless to code a frontend without backend support. Flask lesson are conducted in the last part, limited the coding duration.
3. By using zoom lecture, difficult to gain support through raising questions. Collaboration is unsatisfactory.

Improvement considered

1. Adding a visa gateway during payment
2. Buy and sell fund by clicking a dynamic chart powered by ajax.
3. Adding administration panel to manage user’s activity
4. Create transaction record for users to trace their order record
5. Provide profile photos upload in My Account Page.

Market Value

It depends on the deposit amount from investors, it is illegal to sell funds in Hong Kong without getting approval from Securities and Futures Commission (SFC). If the project is launched, a capital of $20B is expected upon collection.

Business Strategy

Since it is hard to get SFC approval without concrete business activity for the company. The trading business should be register in EU Cyprus regulated by Cypriot Investment Firm (CIF), under the jurisdiction of the Cyprus Securities Exchange Commission (CySEC). The system must also accept visa payment, wire transfer and transaction in terms of US Dollars.

Conclusion

This project reflects the possibility to conduct business in Hong Kong but the operation (Server and customer service) should be enlisted in EU countries, HK can be assessed as a base for management and development due to regulation issues. This prototype has the potential to further develop as a business to serve interested investor who want to invest in different categories of funds.

Code reference

The major logic and process codes are displayed here for reference, comments are added pending for improvement and re-structure.

The code below are only snippets, it does not fully functionable except the version from GitHub.

|  |
| --- |
| Manager.py (The application starts here, insert data into database and create tables) |
| from flask\_script import Manager  # obtain info from views.py and instruct manager.py  from app.views import app  from db import dbhelper  # manager.py manages and prompt command to execute in dbhelper.py  # Preliminary setting: pip install flask-script  # Initialize database(type in terminal): python manager.py create\_tables  # Insert data(type in terminal): python manager.py load\_data  # Start server here: python manager.py runserver  manager = Manager(app)  ################################################################  #Execute shell debug: python manager.py shell                  #  #from app.models import Customer, Goods, Orders, OrderLineItem #  #from app.view import db                                       #  #orders = db.session.query(Orders).filter\_by(id='xxxxx').one() #  #orders.orderLineItems                                         #  ################################################################  # Execute dbhelper createtablesfunction  # @manager.command to invoke flask script act as cmd commands  @manager.command  def create\_tables():      dbhelper.create\_tables()  # Execute dbhelper to insert store data into the database  @manager.command  def load\_data():      dbhelper.load\_data()  if \_\_name\_\_ == '\_\_main\_\_':      manager.run() |

|  |
| --- |
| dbhelper.py (Supports manager.py commands) |
| # create database table  def create\_tables():      # execute create table SQL statement from store-schema.sql      f\_name = 'db/store-schema.sql'      with open(f\_name, 'r', encoding='utf-8') as f:          sql = f.read()          # Establish database connection          conn = sqlite3.connect(DB\_FILES)          try:              # execute database script              conn.executescript(sql)              print('Database initialize success')          except Exception as e:              print('database initialize fail')              print(e)          finally:              # close database connection              conn.close()  # insert record into database  def load\_data():      # execute insert record SQL statement from store-dataload.sql      f\_name = './db/store-dataload.sql'      with open(f\_name, 'r', encoding='utf-8') as f:          sql = f.read()          # Establish database connection          conn = sqlite3.connect(DB\_FILES)          try:              # execute database script              conn.executescript(sql)              print('Data insertion success')          except Exception as e:              print('Data insertion fail')              print(e)          finally:              # close database connection              conn.close() |

|  |
| --- |
| Store-dataload.sql (Record to be inserted into .db) |
| insert into Goods values (1,'Heavy Magnet Dynamic High Income Fund',9000,'Heavy Magnet Dynamic High Income Fund \*\*\*\* ','DT','Heavy Magnet Series','Currency','Mutual','2020-12-31','Hong Kong','Michael Chan','DT.png', 'HKD $11,000');  insert into Goods values (2,'Green Track Mutual Fund',30000,'Green Track Mutual Fund \*\* ','DT','DT Fund Series','Currency','Mutual','2020-12-31','Hong Kong','Derons Chan','DT.png', 'HKD $8,000'); |

|  |
| --- |
| Store-schema.sql (define database structure, support store-dataload.sql to create tables) |
| drop table if exists Customers;  drop table if exists OrderLineItems;  drop table if exists Goods;  drop table if exists Orders;  /\*==============================================================\*/  /\* Table: Customers                                             \*/  /\*==============================================================\*/  create table Customers  (     id                  varchar(20) primary key,     name                 varchar(50) not null,     password             varchar(20) not null,     address              varchar(100),     phone                varchar(20),     birthday             varchar(20)  );  /\*==============================================================\*/  /\* Table: Goods, Funds products                                 \*/  /\*==============================================================\*/  create table Goods  (     id                  integer  primary key autoincrement,     name                varchar(100) not null,     price                float,     description          varchar(200),     brand                varchar(30),     series               varchar(30),     asset\_class          varchar(30),     fund\_type            varchar(30),     maturity             varchar(30),     domicile             varchar(30),     manager              varchar(30),     image                varchar(100),     redem\_amount         float  ); |

|  |
| --- |
| Config.py (Flask Global Configuration) |
| # Enable Flask Debug Mode  DEBUG = True  # Set SQL DBMS Path  SQLALCHEMY\_DATABASE\_URI = 'sqlite:///../db/database.db'  # Enable SQL Output  SQLALCHEMY\_ECHO = True  # Flask Secret key  SECRET\_KEY = 'enydMfghghdghfdgfdfdggpMjf' |

|  |
| --- |
| Views.py (Main code of the application) |
| from flask import Flask, request, session, render\_template, redirect, url\_for, flash  from flask\_sqlalchemy import SQLAlchemy  from app.forms import CustomerRegForm, LoginForm  from app.models import Customer, Goods, Orders, OrderLineItem  import config  import random  import datetime  # Enable flask config  app = Flask(\_\_name\_\_)  # apply config to config.py  app.config.from\_object(config)  # applies SQLAlchemy to db (database)  db = SQLAlchemy(app)  # route to registration page  @app.route('/reg', methods=['GET', 'POST'])  def register():      # Refer to forms.py CustomerRegForm function      form = CustomerRegForm()      if request.method == 'POST':          if form.validate():              # Extract data from forms.py and insert into models.py database Customer(base)              new\_customer = Customer()              new\_customer.id = form.userid.data              new\_customer.name = form.name.data              new\_customer.password = form.password.data              new\_customer.address = form.address.data              new\_customer.birthday = form.birthday.data              new\_customer.phone = form.phone.data              # Call Database SQLAlchemy lib, add record into DBMS              db.session.add(new\_customer)              db.session.commit()              print('registration success')              return render\_template('customer\_reg\_success.html', form=form)      return render\_template('customer\_reg.html', form=form)  @app.route('/login', methods=['GET', 'POST'])  def login():      # Refer to forms.py      form = LoginForm()      if request.method == 'POST':          if form.validate():              c = db.session.query(Customer).filter\_by(id=form.userid.data).first()              if c is not None and c.password == form.password.data:                  print('Login success')                  print('!!! DEBUG c.id !!!', c.id)                  flash('Login success')                  flash('Welcome, ' + c.name)                  customer = {}                  customer['id'] = c.id                  customer['name'] = c.name                  customer['password'] = c.password                  customer['address'] = c.address                  customer['phone'] = c.phone                  customer['birthday'] = c.birthday                  # keep customer http session alive                  session['customer'] = customer                  return redirect(url\_for('main'))              else:                  flash('Wrong UserID or Password, please try again !')                  return render\_template('login.html', form=form)      return render\_template('login.html', form=form)  # Logout the system, delete the session key  @app.route('/logout')  def logout():      if 'customer' not in session.keys():          flash('Please login to access this page')          return redirect(url\_for('login'))      else:          session.clear()          flash('Logout Success')      return redirect(url\_for('login'))  # My account page  @app.route('/account')  def account():      form = LoginForm()      if 'customer' not in session.keys():          flash('Please login to access this page')          print('Logout')          return redirect(url\_for('login'))      # Search goodsid and get details from DBMS DB\_Files      cust = session['customer']      list = []      print('!!! DEBUG cust !!!', cust)          # Item[0] = Fund ID          # Item[1] = Fund name          # Item[2] = Fund Price          # Item[3] = Fund order quantity          # subtotal = Fund total amount      print('!!! DEBUG address !!!', cust['address'])      list.append((cust['id']))      list.append((cust['name']))      list.append((cust['address']))      list.append((cust['phone']))      list.append((cust['birthday']))      print('!!! DEBUG list !!!', list[0])      # Redirect to that product after fetching details from DBMS      return render\_template('account.html', list=list)  # Website Stock Main Page  # Default redirect to home page after typing ip  @app.route('/')  @app.route('/home')  def home():      return render\_template('home.html')  # Shows the home page of the website, provide user option to choose  @app.route('/main')  def main():      # Redirect to login page if not yet logn, session not valid      if 'customer' not in session.keys():          flash('Please login to access this page')          return redirect(url\_for('login'))      return render\_template('main.html') |

|  |
| --- |
| Models.py |
| # Refer to store-schema.sql  # Define customer framework  class Customer(Base):      \_\_tablename\_\_ = 'customers'      id = Column('id', String(20), primary\_key=True)      name = Column('name', String(50), nullable=False)      password = Column('password', String(20), nullable=False)      address = Column('address', String(100))      phone = Column('phone', String(20))      birthday = Column('birthday', String(20)) |

|  |
| --- |
| Forms.py (Used to validate user input) |
| from flask\_wtf import Form  from wtforms import StringField, PasswordField, validators  #WTF is used for data validation  #Need pip install flask-WTF before using it  #Use WTF: xxxField('Username', [validators.required('xxxx')])  #WTForm types:  #StringField-> only string is ok  #IntegerField-> only integers  #DecimalField  #PasswordField  #BooleanField  #RadioField  #SelectField  #TextAreaField  #SubmitField  #Validators info:  #DataRequired: Check if empty  #Email: Check valid email  #IPAddress: Check valid IP Address  #Length: password/ text liength is within expectation  #NumberRange: Set Number Range  #URL: Validate URL  #EqualTo: Check if the content is the same  class LoginForm(Form):      '''User Login Form validation'''      userid = StringField('UserID：', [validators.DataRequired('Username field is required')])      password = PasswordField('Password：', [validators.DataRequired('Password is required')]) |

|  |
| --- |
| CSS Script |
| div.header {      text-align: center;      font-size: 1.5em;      padding-top: 1.0em;      padding-bottom: 1.0em;      background-color: #E8E8E8;      border: solid 1px;  }  a:link {      font-size: 14px;      color: #333333;      text-decoration: none;  }  a:visited {      color: #333333;      text-decoration: none;  }  a:hover {      color: #0066CC;  } |

|  |
| --- |
| Base1.html (Template for html, error case) |
| <!-- Providing web template for main index page -->  <!-- Block template starts with {percentage percentage} -->  <!-- base2 change class to success -->  <html>      <head>          <meta charset="utf-8">          <!-- Block template show the page title -->           <!-- Bootstrap CSS -->           <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css" integrity="sha384-Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh" crossorigin="anonymous">             <title>              {% block title %}{% endblock %}          </title>          <!-- url\_for static is to handle external file -->          <link rel="stylesheet" type="text/css" href="{{ url\_for('static', filename='css/public.css') }}">          <link rel="stylesheet" type="text/css" href="{{ url\_for('static', filename='css/style.css') }}">      </head>      <body>          <!-- Header Info -->          <div class="header">DT Financial Service</div>          <hr width="100%" />          <!-- Display error message -->         {% with messages = get\_flashed\_messages(with\_categories=true) %}         {% if messages %}         <br>            <ul>                {% for category, message in messages %}                <div class="alert alert-danger alert-dismissible" role="alert">                    <button type="button" class="close" data-dismiss="alert" aria-label="Close"><span aria-hidden="true">&times;</span></button>                    <!-- <strong>Title</strong> --> {{ message }}                    </div>               {% endfor %}            </ul>          {% endif %}          {% endwith %}          <!-- Main Content Info -->          {% block body %}{% endblock %} |

|  |
| --- |
| Base2.html (Template for html, successful case) |
| <!-- Providing web template for main index page -->  <!-- Block template starts with {percentage percentage} -->  <!-- base2 change class to success -->  <html>      <head>          <meta charset="utf-8">          <!-- Block template show the page title -->           <!-- Bootstrap CSS -->           <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css" integrity="sha384-Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh" crossorigin="anonymous">             <title>              {% block title %}{% endblock %}          </title>          <!-- url\_for static is to handle external file -->          <link rel="stylesheet" type="text/css" href="{{ url\_for('static', filename='css/public.css') }}">          <link rel="stylesheet" type="text/css" href="{{ url\_for('static', filename='css/style.css') }}">      </head>      <body>          <!-- Header Info -->          <div class="header">DT Financial Service</div>          <hr width="100%" />          <!-- Display error message -->         {% with messages = get\_flashed\_messages(with\_categories=true) %}         {% if messages %}         <br>            <ul>                {% for category, message in messages %}                <div class="alert alert-danger alert-dismissible" role="alert">                    <button type="button" class="close" data-dismiss="alert" aria-label="Close"><span aria-hidden="true">&times;</span></button>                    <!-- <strong>Title</strong> --> {{ message }}                    </div>               {% endfor %}            </ul>          {% endif %}          {% endwith %} |

|  |
| --- |
| Home.html (Frontend Root of website) |
| {% extends "base2.html" %}  {% block title %}Home{% endblock %}  {% block body %}  <!doctype html>  <html lang="en">    <head>      <!-- Required meta tags -->      <meta charset="utf-8">      <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">      <!-- Bootstrap CSS -->      <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css" integrity="sha384-Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh" crossorigin="anonymous">      <!-- Custom CSS Style -->      <style>      #section1 {padding-top:50px;height:500px;color: #fff; background-color: #1E88E5;}      #section2 {padding-top:50px;height:500px;color: #fff; background-color: #673ab7;}      .background{          background-image: url("http://dtgroups.ddns.net/dthrms/wp-content/uploads/2020/03/sea-edge-79ab30e2.png");          background-color: aquamarine;          background-position: left center;          -webkit-background-size: cover;          -moz-background-size: cover;          -o-background-size: cover;          background-size: cover;      }      .glow-on-hover {      width: 220px;      height: 50px;      border: none;      outline: none;      color: #fff;      background: #111;      cursor: pointer;      position: relative;      z-index: 0;      border-radius: 10px;      }      .glow-on-hover:before {          content: '';          background: linear-gradient(45deg, #ff0000, #ff7300, #fffb00, #48ff00, #00ffd5, #002bff, #7a00ff, #ff00c8, #ff0000);          position: absolute;          top: -2px;          left:-2px;          background-size: 400%;          z-index: -1;          filter: blur(5px);          width: calc(100% + 4px);          height: calc(100% + 4px);          animation: glowing 20s linear infinite;          opacity: 0;          transition: opacity .3s ease-in-out;          border-radius: 10px;      }      .glow-on-hover:active {          color: #000      }  <!-- Content Section 1 -->              <div id="section1" class="container-fluid">                  <h3 class="my-3">DT Investment Management </h3>                  <p>Creativity bounds us together !</p>                  <br>                  <p>Sign up to subscribe our weekly Project.</p>                  <p style="font-weight: 400; font-family: Georgia, 'Times New Roman', Times, serif;">The term “Fund” covers a number of terms including mutual funds, collective                      investment undertakings, collective investment schemes or pooled investment                      vehicles and these terms are generally used interchangeably.                      A fund can be classified in various ways e.g. by its structure, its investment                      strategy or its regulatory status.</p>                      <br><br>                      <button class="glow-on-hover" type="button" onclick="location.href='/login'"> Login </button>              </div>          </div>          <!-- /.row -->          <!-- Content Section 2 -->          <div id="section2" class="row container-fluid">              <div class="col-lg-3 col-md-6 mb-4">                  <div class="card h-100">                    <img class="card-img-top" src="http://dtgroups.ddns.net/dthrms/wp-content/uploads/2020/04/today-5-18-500x325.png" alt="">                    <div class="card-body">                      <h4 class="card-title" style="color: black;">About DT</h4>                      <p class="card-text" style="color: black;">Discover DT Group and grap a job !</p>                    </div>                    <div class="card-footer">                      <a href="http://dtgroups.ddns.net/dthrms/about-us/" class="btn btn-primary">Find Out More!</a>                    </div>                  </div>                </div>                  <div class="col-lg-3 col-md-6 mb-4">                  <div class="card h-100">                    <img class="card-img-top" src="http://dtgroups.ddns.net/dthrms/wp-content/uploads/2020/04/quiet-breaks-with-god-500x325.jpg" alt="">                    <div class="card-body">                      <h4 class="card-title" style="color: black;">Register Account</h4>                      <p class="card-text" style="color: black;">Register DT Fund to start your fund career.</p>                    </div>                    <div class="card-footer">                      <a href="/reg" class="btn btn-primary">Register</a>                    </div>                  </div>                </div>  <!-- JavaScript Section -->          <!-- jQuery first, then Popper.js, then Bootstrap JS -->          <script src="https://code.jquery.com/jquery-3.4.1.slim.min.js" integrity="sha384-J6qa4849blE2+poT4WnyKhv5vZF5SrPo0iEjwBvKU7imGFAV0wwj1yYfoRSJoZ+n" crossorigin="anonymous"></script>          <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js" integrity="sha384-Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo" crossorigin="anonymous"></script>          <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js" integrity="sha384-wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4Ih7YwaYd1iqfktj0Uod8GCExl3Og8ifwB6" crossorigin="anonymous"></script>          <script type="text/javascript" src="https://s3.tradingview.com/tv.js"></script>            <!-- Stock graph -->          <script type="text/javascript">          new TradingView.widget(              {              "width": 695,              "height": 455,              "symbol": "INDEX:HSI",              "interval": "D",              "timezone": "Etc/UTC",              "theme": "light",              "style": "1",              "locale": "en",              "toolbar\_bg": "#f1f3f6",              "enable\_publishing": false,              "allow\_symbol\_change": true,              "container\_id": "tradingview\_57e42"              }          );          </script>          <!-- Clock -->          <script>              var canvas = document.getElementById("canvas");              var ctx = canvas.getContext("2d");              var radius = canvas.height / 2;              ctx.translate(radius, radius);              radius = radius \* 0.9              setInterval(drawClock, 1000);                function drawClock() {                drawFace(ctx, radius);                drawNumbers(ctx, radius);                drawTime(ctx, radius);              }                function drawFace(ctx, radius) {                var grad;                ctx.beginPath();                ctx.arc(0, 0, radius, 0, 2\*Math.PI);                ctx.fillStyle = 'white';                ctx.fill();                grad = ctx.createRadialGradient(0,0,radius\*0.9, 0,0,radius\*1.05);                grad.addColorStop(0, '#333');                grad.addColorStop(0.5, 'white');                grad.addColorStop(1, '#333');                ctx.strokeStyle = grad;                ctx.lineWidth = radius\*0.1;                ctx.stroke();                ctx.beginPath();                ctx.arc(0, 0, radius\*0.1, 0, 2\*Math.PI);                ctx.fillStyle = '#333';                ctx.fill();              } |

|  |
| --- |
| Login.html (Frontend login page) |
| {% extends "base1.html" %}  {% block title %}User Login{% endblock %}  {% block body %}  <!-- Display login error info -->  <ul>      {% for field, errors in form.errors.items() %}          {% for message in errors %}              <li class="error">{{ message }}</li>          {% endfor %}      {% endfor %}  </ul>  <style>  .background{      background-image: url("http://dtgroups.ddns.net/dthrms/wp-content/uploads/2020/04/pngtree-financial-stock-market-background-illustration-on-technology-abstract-background-image\_322157.jpg");      background-color: aquamarine;      background-position: left center;      -webkit-background-size: cover;      -moz-background-size: cover;      -o-background-size: cover;      background-size: cover;  }  body {      font-family: 'Roboto', sans-serif;      background-size: cover;  }  <!-- Page Content -->  <form action="/login" method="POST">      <!-- Prevent CSRF attack measure -->      {{ form.hidden\_tag() }}      <!-- Login Module -->      <body class="background" data-spy="scroll" data-target=".navbar" data-offset="50">          <div class="modal-dialog text-center">              <div class="col-sm-8 main-section">                  <div class="modal-content">                      <div class="col-12 user-img">                          <img src="{{ url\_for('static', filename='images/face.png') }}">                      </div>                      <form class="col-12">                          <div class="form-group">                              <input type="text" class="form-control" placeholder="Enter ID" {{ form.userid }}                          </div>                          <div class="form-group">                              <input type="password" class="form-control" placeholder="Enter Password" {{ form.password }}                          </div>                          <button type="submit" class="btn" onclick="document.forms[0].fn.value='login'"><i class="fas fa-sign-in-alt"></i>Login</button>                          <a href="/reg"></a>                      </form>                      <div class="col-12 forgot">                          <a href="/home">Back to Home Page</a>                      </div>                  </div> <!-- End of Modal Content--> |

|  |
| --- |
| Goods\_list.html (Show the list of fund products) |
| {% extends "base2.html" %}  {% block title %}Fund Subscription{% endblock %}  {% block body %}  <style type="text/css">      table {          border-collapse: collapse;      }      .col1 {          padding-top: 5px;          border-top: 1px dashed #00494e;          text-indent: 40px      }  <hr width="100%"/>  <div class="text3" align="center">Please choose your interested fund from the below list</div>  <br>  <h4 style="text-align: center;">Risk Disclosure:</h4>  <div class="text2" align="center">The financial products offered by the company carry a      high level of risk and can result in the loss of all your funds. You should never      invest money that you cannot afford to lose. Before making investment decisions,      investors should carefully consider whether investment products/ services      are suitable in light of their financial position, investment      objectives and experiences, risk tolerance and other      relevant circumstances.</div>  <br>  <table class="table table-hover" style="text-align: center;" width="100%" border="0" align="center">      <tr bgcolor="#b4c8ed">          <th>Fund name</th>          <th width="5%">Fund Market Value</th>          <th width="15%">Add to cart</th>      </tr>    {% for goods in list %}      <tr bgcolor={{ loop.cycle('#ffffff', '#edf8ff') }}>          <td class="col1"><a href="/detail?id={{ goods.id }}">{{ goods.description }}</a></td>          <td class="col2">HKD ${{ goods.price}}</td>          <td class="col3"><a href="/add?id={{ goods.id }}&name={{ goods.name }}&price={{ goods.price }}">Subscribe</a></td>      </tr>    {% endfor %}  </table>  </body>  {% endblock %} |