受理号: ________________　受理签字: ________________

登记号: ________________　审查签字: ________________

流水号	****************

计算机软件著作权登记申请表

软件基本信息	软件全称	Hollow 体育中心管理系统软件	版本号	V1.0
	软件简称		分类号	30106-9100
	软件作品说明	[✓ 原创] ○ 修改(含翻译软件、合成软件) □ 修改软件须经原权利人授权 □ 原有软件已经登记 • 原登记号: • 修改（翻译或合成）软件作品说明:		
开发完成日期		2020 年 05 月 15 日		
发表状态		○ 已发表 [✓ 未发表]		
开发方式		[✓ 独立开发] ○ 合作开发 ○ 委托开发 ○ 下达任务开发		

	姓名或名称	类别	证件类型	证件号码	国籍	省份/城市	园区
著作权人	Hollow Man	自然人	居民身份证	********************	中国	*****************	

流水号	******************

权利说明	权利取得方式	⊙ 原始取得 ○ 继受取得（ ○ 受让 ○ 承受 ○ 继承 ） ▁ 该软件已登记 （原登记号：__） □ 原登记做过变更或补充 （变更或补充证明编号：__）		
	权利范围	⊙ 全部 ○ 部分（ □ 发表权 □ 署名权 □ 修改权 □ 复制权 □ 发行权 □ 出租权 □ 信息网络传播权 □ 翻译权 □ 应当由著作权人享有的其他权利 ）		
软件鉴别材料	⊙ 一般交存	提交源程序前连续的30页和后连续的30页； 提交任何一种文档的前连续的30页和后连续的30页； ⊙ 一种文档 ○ ____种文档		
	○ 例外交存	○使用黑色宽斜线覆盖，页码为： ○前10页和任选连续的50页 ○目标程序的连续的前、后各30页和源程序任选连续的20页		
软件功能和技术特点	硬件环境	CPU: Intel Core i7-8550U 1.80GHZ， RAM: 8GB，硬盘: SSD 128GB 机械 1TB		
	软件环境	Ubuntu 18.04,Python 3.7.5,django-qrcode,alembic,blinker,click,dominate,Flask,Flask-Bootstrap,Flask-Login,Flask-Mail,Flask-Migrate,Flask-Moment,Flask-SQLAlchemy,Flask-WTF,itsdangerous,Jinja2,Mako,MarkupSafe,python-dateutil,python-editor,SQLAlchemy,visitor		
	编程语言	Python 3, HTML, CSS	源程序量	3158
	主要功能和技术特点	使用Python Flask实现了一个体育中心管理系统，该系统可以查看指定时间/天数范围内所有设施的时间表，查看指定设施在一定时间/天数范围内的时间表，支付每月或每年的会员费，在指定的日期和时间预订并支付活动费用，取消会员资格，取消预订，展示设施图片，显示设施的定价列表，银行卡预定付款（模拟），存储收据，并按需显示，生成PDF收据，可以显示收据二维码，用于验证，支持用户帐户和用户登录，存储用户的支付卡详细信息，以便快速结账，用户帐户和数据具有良好的安全性，可以配置设施和活动，此管理系统软件提供了响应灵敏、友好的用户界面。		

流水号	******************

申请办理方式		⊙由著作权人申请　○由代理人申请		
申请人信息	姓名或名称	Hollow Man	电话	************
	详细地址	**	邮编	******
	联系人	Hollow Man	手机	*************
	E-mail	******************	传真	
代理人信息	申请人委托下述代理人办理登记事宜，具体委托事项如下：			
	姓名或名称		电话	
	详细地址		邮编	
	联系人		手机	
	E-mail		传真	

申请人认真阅读了填表说明，准确理解了所需填写的内容，保证所填写的内容真实。

申请人签章：

2020 年 05 月 25 日

流水号	2020R11L664853

证书份数	1份正本
请确认所需要的计算机软件著作权登记证书副本份数。登记证书正本和副本数量之和不能超过软件著作权人的数量。	

提交申请材料清单

申请材料类型	**申请材料名称**	
申请表	打印签字或盖章的登记申请表	一份 4 页
软件鉴别材料	软件源程序	一份 60 页
	软件文档(1)	一份 26 页
	软件文档(2)	一份____页
身份证明文件	申请人身份证明复印件	一份 1 页
	代理人身份证明复印件	一份____页
权利归属证明文件	软件转让合同或协议	一份____页
	承受或继承证明文件	一份____页
其他材料		一份____页
		一份____页
		一份____页
		一份____页

填写说明：

请按照提示要求提交有关申请材料，并在提交申请材料清单中准确填写实际交存材料页数。若提示中没有的，请填写材料名称及其页数。该页是申请表的组成部分与申请表一并打印提交。

```python
# config.py

import os
basedir = os.path.abspath(os.path.dirname(__file__))

class Config:
    SECRET_KEY = os.environ.get('SECRET_KEY') or 'hard to guess string'
    MAIL_SERVER = os.environ.get('MAIL_SERVER', '')
    MAIL_PORT = int(os.environ.get('MAIL_PORT', '587'))
    MAIL_USE_TLS = os.environ.get('MAIL_USE_TLS', 'true').lower() in \
        ['true', 'on', '1']
    MAIL_USERNAME = ''
    MAIL_PASSWORD = ''
    FLASKY_MAIL_SUBJECT_PREFIX = '[Flasky]'
    FLASKY_MAIL_SENDER = ''
    FLASKY_ADMIN = os.environ.get('FLASKY_ADMIN')
    SQLALCHEMY_TRACK_MODIFICATIONS = False

    @staticmethod
    def init_app(app):
        pass

class DevelopmentConfig(Config):
    DEBUG = True
    SQLALCHEMY_DATABASE_URI = os.environ.get('DEV_DATABASE_URL') or \
        'sqlite:///' + os.path.join(basedir, 'data-dev.sqlite')

class TestingConfig(Config):
    TESTING = True
    SQLALCHEMY_DATABASE_URI = os.environ.get('TEST_DATABASE_URL') or \
        'sqlite://'

class ProductionConfig(Config):
    SQLALCHEMY_DATABASE_URI = os.environ.get('DEV_DATABASE_URL') or \
        'sqlite:///' + os.path.join(basedir, 'data-dev.sqlite')
    #use ProductionConfig to initialize app
    @classmethod
    def init_app(cls, app):
        Config.init_app(app)

        # email errors to the administrators
        # setup logger
        # import logging
        # from logging.handlers import SMTPHandler
        # credentials = None
        # secure = None
        # if getattr(cls, 'MAIL_USERNAME', None) is not None:
        #     credentials = (cls.MAIL_USERNAME, cls.MAIL_PASSWORD)
        #     if getattr(cls, 'MAIL_USE_TLS', None):
        #         secure = ()
        # mail_handler = SMTPHandler(
        #     mailhost=(cls.MAIL_SERVER, cls.MAIL_PORT),
        #     fromaddr=cls.FLASKY_MAIL_SENDER,
        #     toaddrs=[cls.FLASKY_ADMIN],
        #     subject=cls.FLASKY_MAIL_SUBJECT_PREFIX + ' Application Error',
        #     credentials=credentials,
        #     secure=secure)
        # #only send Error message to admin
        # mail_handler.setLevel(logging.ERROR)
        # app.logger.addHandler(mail_handler)

        import logging
        # setup flask log
        app.debug=True
        handler = logging.FileHandler('flask.log', encoding='UTF-8')
        handler.setLevel(logging.INFO)
        logging_format = logging.Formatter(
            '%(asctime)s - %(levelname)s - %(filename)s - %(funcName)s - %(lineno)s - %(message)s')
        handler.setFormatter(logging_format)
        app.logger.addHandler(handler)


config = {
    'development': DevelopmentConfig,
    'testing': TestingConfig,
    'production': ProductionConfig,

    'default': DevelopmentConfig
}

# flasky.py

import os
import click
from flask_migrate import Migrate
from app import create_app, db
from app.models import User, Role, Permission,Membership_type,Membership,Facility,Credit_card_info,Activity,Account,Booking,Time_management

app = create_app(os.getenv('FLASK_CONFIG') or 'default')
migrate = Migrate(app, db)

```

```
@app.shell_context_processor
def make_shell_context():
    return dict(db=db, User=User, Role=Role, Permission=Permission,Membership_type=Membership_type,Membership=Membership,
                   Facility=Facility,Booking=Booking,Time_management=Time_management,Activity=Activity,Credit_card_info=Credit_card_info,
                   Account=Account)


@app.cli.command()
@click.argument('test_names', nargs=-1)
def test(test_names):
    """Run the unit tests."""
    import unittest
    if test_names:
        tests = unittest.TestLoader().loadTestsFromNames(test_names)
    else:
        tests = unittest.TestLoader().discover('tests')
    unittest.TextTestRunner(verbosity=2).run(tests)

# app/__init__.py

from flask import Flask
from flask_bootstrap import Bootstrap
from flask_mail import Mail
from flask_moment import Moment
from flask_sqlalchemy import SQLAlchemy
from flask_login import LoginManager
from config import config

bootstrap = Bootstrap()
mail = Mail()
moment = Moment()
db = SQLAlchemy()

login_manager = LoginManager()
login_manager.login_view = 'auth.login'


def create_app(config_name):
    app = Flask(__name__)
    app.config.from_object(config[config_name])
    config[config_name].init_app(app)

    bootstrap.init_app(app)
    mail.init_app(app)
    moment.init_app(app)
    db.init_app(app)
    login_manager.init_app(app)

```

```python
    from .main import main as main_blueprint
    app.register_blueprint(main_blueprint)

    from .auth import auth as auth_blueprint
    app.register_blueprint(auth_blueprint, url_prefix='/auth')

    return app

# app/decorators.py

from functools import wraps
from flask import abort
from flask_login import current_user
from .models import Permission

def permission_required(permission):
    def decorator(f):
        @wraps(f)
        def decorated_function(*args, **kwargs):
            if not current_user.can(permission):
                abort(403)
            return f(*args, **kwargs)
        return decorated_function
    return decorator

def admin_required(f):
    return permission_required(Permission.ADMIN)(f)

# app/email.py

from threading import Thread
from flask import current_app, render_template
from flask_mail import Message
from . import mail
import pdfkit

def send_async_email(app, msg):
    with app.app_context():
        mail.send(msg)

def send_email(to, subject, template, **kwargs):
    app = current_app._get_current_object()
    msg = Message(app.config['FLASKY_MAIL_SUBJECT_PREFIX'] + ' ' + subject,
                  sender=app.config['FLASKY_MAIL_SENDER'], recipients=[to])
    msg.body = render_template(template + '.txt', **kwargs)
    msg.html = render_template(template + '.html', **kwargs)
    thr = Thread(target=send_async_email, args=[app, msg])
    thr.start()
    return thr

def str_to_pdf(string, to_file):
    # 将 wkhtmltopdf.exe 程序绝对路径传入 config 对象
    path_wkthmltopdf = r'C:\\Program Files\\wkhtmltopdf\\bin\\wkhtmltopdf.exe'
    config = pdfkit.configuration(wkhtmltopdf=path_wkthmltopdf)
    # 生成 pdf 文件，to_file 为文件路径
    pdfkit.from_string(string, to_file, configuration=config)
    print('完成')


# app/models.py

from datetime import datetime, timedelta
import hashlib
from werkzeug.security import generate_password_hash, check_password_hash
from itsdangerous import TimedJSONWebSignatureSerializer as Serializer
from flask import current_app, request
from flask_login import UserMixin, AnonymousUserMixin
from . import db, login_manager

class Permission:
    PAY= 1
    CANCEL = 2
    HANDLE = 3
    DISPLAY = 4
    VIEW_BUSINESS = 5
    CONFIGURE = 6
    MODERATE = 8
    ADMIN = 16

class Role(db.Model):
    __tablename__ = 'roles'
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(64), unique=True)
    default = db.Column(db.Boolean, default=False, index=True)
    permissions = db.Column(db.Integer)
    users = db.relationship('User', backref='role', lazy='dynamic')

    def __init__(self, **kwargs):
        super(Role, self).__init__(**kwargs)
        if self.permissions is None:
            self.permissions = 0

    @staticmethod
    def insert_roles():
        roles = {
            'User': [Permission.PAY, Permission.CANCEL, Permission.HANDLE,Permission.DISPLAY],
            'Employee':[Permission.PAY, Permission.CANCEL, Permission.HANDLE,Permission.DISPLAY],
            'Manager': [Permission.VIEW_BUSINESS, Permission.CONFIGURE],

            'Moderator': [ Permission.MODERATE],
            'Administrator': [ Permission.MODERATE,Permission.ADMIN],

        }
        default_role = 'User'
        for r in roles:
            role = Role.query.filter_by(name=r).first()
            if role is None:
                role = Role(name=r)
            role.reset_permissions()
            for perm in roles[r]:
                role.add_permission(perm)
            role.default = (role.name == default_role)
            db.session.add(role)
        db.session.commit()

    def add_permission(self, perm):
        if not self.has_permission(perm):
            self.permissions += perm

    def remove_permission(self, perm):
        if self.has_permission(perm):
            self.permissions -= perm

    def reset_permissions(self):
        self.permissions = 0

    def has_permission(self, perm):
        return self.permissions & perm == perm

    def __repr__(self):
        return '<Role %r>' % self.name

class Membership_type(db.Model):
    __tablename__ = 'membership_type'
    id=db.Column(db.Integer,primary_key=True)
    length=db.Column(db.Integer,nullable=False)
    price=db.Column(db.Float,nullable=False)
    memberships = db.relationship('Membership',backref='user',lazy='dynamic')

class Membership(db.Model): #have a one-to-one relationship with user since one user can only have at most one membership
    __tablename__ = 'membership'
    id=db.Column(db.Integer,primary_key=True)
    title = db.Column(db.String(8), index=True)
    firstname = db.Column(db.String(32), index=True)
    lastname = db.Column(db.String(32), index=True)
    status = db.Column(db.String(16), nullable=False)
    start_date=db.Column(db.DateTime, default=datetime.utcnow)
    end_date=db.Column(db.DateTime, default=datetime.utcnow)
    membership_type_id=db.Column(db.Integer,db.ForeignKey('membership_type.id'))
    account_id=db.Column(db.Integer,db.ForeignKey('accounts.id'))

    def valueOfEnd_date(self,length):
        self.end_date = self.start_date + timedelta(days=length)
        db.session.add(self)

class User(UserMixin, db.Model): #customers
    __tablename__ = 'users'
    id = db.Column(db.Integer, primary_key=True)
    email = db.Column(db.String(64), unique=True, index=True)
    username = db.Column(db.String(64), unique=True, index=True)
    role_id = db.Column(db.Integer, db.ForeignKey('roles.id'),default=1)
    password_hash = db.Column(db.String(128))
    confirmed = db.Column(db.Boolean, default=False)

    def __init__(self, **kwargs):
        super(User, self).__init__(**kwargs)
        if self.role is None:
            if self.email == current_app.config['FLASKY_ADMIN']:
                self.role = Role.query.filter_by(name='Administrator').first()
            if self.role is None:
                self.role = Role.query.filter_by(default=True).first()

    @property
    def password(self):
        raise AttributeError('password is not a readable attribute')

    @password.setter
    def password(self, password):
        self.password_hash = generate_password_hash(password)

    def verify_password(self, password):
        return check_password_hash(self.password_hash, password)

    def generate_confirmation_token(self, expiration=3600):
        s = Serializer(current_app.config['SECRET_KEY'], expiration)
        return s.dumps({'confirm': self.id}).decode('utf-8')

    def confirm(self, token):
        s = Serializer(current_app.config['SECRET_KEY'])
        try:
            data = s.loads(token.encode('utf-8'))
        except:
            return False
        if data.get('confirm') != self.id:
            return False
        self.confirmed = True
        db.session.add(self)
        return True

    def generate_reset_token(self, expiration=3600):
        s = Serializer(current_app.config['SECRET_KEY'], expiration)
        return s.dumps({'reset': self.id}).decode('utf-8')

    @staticmethod
    def reset_password(token, new_password):
        s = Serializer(current_app.config['SECRET_KEY'])
        try:
            data = s.loads(token.encode('utf-8'))
        except:
            return False
        user = User.query.get(data.get('reset'))
        if user is None:
            return False
        user.password = new_password
        db.session.add(user)
        return True

    def generate_email_change_token(self, new_email, expiration=3600)
:
        s = Serializer(current_app.config['SECRET_KEY'], expiration)
        return s.dumps(
            {'change_email': self.id, 'new_email': new_email}).decode
('utf-8')

    def change_email(self, token):
        s = Serializer(current_app.config['SECRET_KEY'])
        try:
            data = s.loads(token.encode('utf-8'))
        except:
            return False
        if data.get('change_email') != self.id:
            return False
        new_email = data.get('new_email')
        if new_email is None:
            return False
        if self.query.filter_by(email=new_email).first() is not None:
            return False
        self.email = new_email
        db.session.add(self)
        return True

    def can(self, perm):
        return self.role is not None and self.role.has_permission(perm)

    def is_administrator(self):
        return self.can(Permission.ADMIN)

    def ping(self):
        self.last_seen = datetime.utcnow()
        db.session.add(self)

    def gravatar(self, size=100, default='identicon', rating='g'):
        url = 'https://secure.gravatar.com/avatar'
        hash = hashlib.md5(self.email.lower().encode('utf-8')).hexdigest()
        return '{url}/{hash}?s={size}&d={default}&r={rating}'.format(
            url=url, hash=hash, size=size, default=default, rating=rating)

    def __repr__(self):
        return '<User %r>' % self.username

class AnonymousUser(AnonymousUserMixin):
    def can(self, permissions):
        return False

    def is_administrator(self):
        return False

login_manager.anonymous_user = AnonymousUser

@login_manager.user_loader
def load_user(user_id):
    return User.query.get(int(user_id))

"""
```

```
#each staff can manage many facilities and each facility can be managed by many staffs.
manage_facility_table=db.Table('manage_facility_table',
                               db.Column('staff_id', db.Integer, db.ForeignKey('staff.id'), primary_key=True),
                               db.Column('facility_id', db.Integer, db.ForeignKey('facility.id'), primary_key=True))
                               """


class Facility(db.Model): #facility table, each facility can have many courts and activities
    __tablename__ = 'facilities'
    id=db.Column(db.Integer,primary_key=True)
    name=db.Column(db.String(16),nullable=False)
    url = db.Column(db.String(256), nullable=False)
    capacity=db.Column(db.Integer)
    description=db.Column(db.String(64),nullable=False)
    times = db.relationship('Time_management', backref='court',
                            lazy='dynamic')  # each court have many intervals for booking
    '''courts=db.relationship('Court',backref='facility')'''
    activities=db.relationship('Activity',backref='facility')

class Booking(db.Model): #each user can have many bookings
    __tablename__ = 'bookings'
    id=db.Column(db.Integer,primary_key=True)
    number=db.Column(db.Integer)
    activity=db.Column(db.String(16))
    activity_id = db.Column(db.Integer,db.ForeignKey('activities.id'))
    status=db.Column(db.String(16),nullable=False)
    payment=db.Column(db.String(32),default='Cash')
    fees=db.Column(db.Integer,nullable=False)
    timestamp = db.Column(db.DateTime, default=datetime.utcnow)
    time_id = db.Column(db.Integer, db.ForeignKey("timetables.id"))
    account_id=db.Column(db.Integer,db.ForeignKey('accounts.id'))

class Time_management(db.Model):
    __tablename__ = 'timetables'
    id=db.Column(db.Integer,primary_key=True)
    date=db.Column(db.Date,nullable=False)
    start_time = db.Column(db.Integer,nullable=False)
    end_time = db.Column(db.Integer,nullable=False)
    current_capacity = db.Column(db.Integer,default=0)
    '''booking_id=db.Column(db.Integer,db.ForeignKey(Booking.id),default=-1)'''
    bookings=db.relationship('Booking',backref="timetable")
    facility_id=db.Column(db.Integer,db.ForeignKey(Facility.id))
```

```

"""
class Court(db.Model): #e.g like basketball court, there are 7 courts for use
    __tablename__ = 'courts'
    id=db.Column(db.Integer,primary_key=True)
    number=db.Column(db.Integer)
    availablity=db.Column(db.String,default='available')
    times=db.relationship('Time_management',backref='court',lazy='dynamic') #each court have many intervals for booking
    facility_id = db.Column(db.Integer, db.ForeignKey('facilities.id'))
"""

class Activity(db.Model): #avtivity
    __tablename__ = 'activities'
    id = db.Column(db.Integer, primary_key=True)
    weekly_income = db.Column(db.Integer)
    weekly_usage = db.Column(db.Integer)
    #activity_staff_id=db.Column(db.Integer)
    activity_price=db.Column(db.Integer)
    activity_name=db.Column(db.String(16))
    repeat_every = db.Column(db.Integer)
    bookings = db.relationship('Booking', backref="activity_ii")
    facility_id=db.Column(db.Integer,db.ForeignKey('facilities.id'))

class Credit_card_info(db.Model): #paying card information
    __tablename__ = 'cards'
    id=db.Column(db.Integer,primary_key=True)
    card_number=db.Column(db.Integer,nullable=False)
    expire_month=db.Column(db.Integer,nullable=False)
    expire_year=db.Column(db.Integer,nullable=False)
    security_code=db.Column(db.Integer,nullable=False)
    holder_name=db.Column(db.String(64),nullable=False)
    account_id = db.Column(db.Integer, db.ForeignKey('accounts.id'))

class Account(db.Model):
    __tablename__='accounts'
    id=db.Column(db.Integer, primary_key=True, nullable=False)
    user_id=db.Column(db.Integer,db.ForeignKey('users.id'), unique=True)
    membership = db.relationship('Membership', uselist=False)
    cards = db.relationship('Credit_card_info', backref='account', lazy='dynamic')
    bookings = db.relationship('Booking', backref='account', lazy='dynamic')

"""
class Staff(db.Model):
    __tablename__ = 'staffs'
    id=db.Column(db.Integer,primary_key=True)
    name=db.Column(db.String(64),nullable=False)
    role=db.Column(db.String(64),nullable=False)
    facilities = db.relationship('Facility',secondary=manage_facility
_table,backref='Staff')
    """

"""
class Business(db.Model):
    __tablename__= 'business'
    id = db.Column(db.Integer,primart_key=True)
    people = db.Column(db.Integer)
    income = db.Column(db.Integer)
"""

# app/auth/__init__.py

from flask import Blueprint
auth = Blueprint('auth', __name__)
from . import views

# app/auth/views.py

from flask import render_template, redirect, request, url_for, flash
from flask_login import login_user, logout_user, login_required, \
    current_user
from . import auth
from .. import db
from ..models import User, Account
from ..email import send_email
from .forms import LoginForm, RegistrationForm, ChangePasswordForm,\
    PasswordResetRequestForm, PasswordResetForm, ChangeEmailForm


@auth.before_app_request
def before_request():
    if current_user.is_authenticated:
        current_user.ping()
        if not current_user.confirmed \
                and request.endpoint \
                and request.blueprint != 'auth' \
                and request.endpoint != 'static':
            return redirect(url_for('auth.unconfirmed'))


@auth.route('/unconfirmed')
def unconfirmed():
    if current_user.is_anonymous or current_user.confirmed:
        return redirect(url_for('main.index'))
    return render_template('auth/unconfirmed.html')

@auth.route('/login', methods=['GET', 'POST'])
def login():
    form = LoginForm()
    if form.validate_on_submit():
        user = User.query.filter_by(email=form.email.data.lower()).first()
        if user is not None and user.verify_password(form.password.data):
            login_user(user, form.remember_me.data)
            next = request.args.get('next')
            if next is None or not next.startswith('/'):
                next = url_for('main.index')
            return redirect(next)
        flash('Invalid email or password.')
    return render_template('auth/login.html', form=form)

@auth.route('/logout')
@login_required
def logout():
    logout_user()
    flash('You have been logged out.')
    return redirect(url_for('main.index'))

@auth.route('/register', methods=['GET', 'POST'])
def register():
    form = RegistrationForm()
    if form.validate_on_submit():
        user = User(email=form.email.data.lower(),
                    username=form.username.data,
                    password=form.password.data)
        db.session.add(user)
        db.session.commit()
        token = user.generate_confirmation_token()
        send_email(user.email, 'Confirm Your Account',
                   'auth/email/confirm', user=user, token=token)
        flash('A confirmation email has been sent to you by email.')
        return redirect(url_for('auth.login'))
    return render_template('auth/register.html', form=form)

@auth.route('/confirm/<token>')
@login_required
def confirm(token):
    if current_user.confirmed:
        return redirect(url_for('main.index'))
    if current_user.confirm(token):
        db.session.commit()
        account = Account(user_id=current_user.id)
        db.session.add(account)
        db.session.commit()
        flash('You have confirmed your account. Thanks!')
    else:
        flash('The confirmation link is invalid or has expired.')
    return redirect(url_for('main.index'))

@auth.route('/confirm')
@login_required
def resend_confirmation():
    token = current_user.generate_confirmation_token()
    send_email(current_user.email, 'Confirm Your Account',
               'auth/email/confirm', user=current_user, token=token)
    flash('A new confirmation email has been sent to you by email.')
    return redirect(url_for('main.index'))

@auth.route('/change-password', methods=['GET', 'POST'])
@login_required
def change_password():
    form = ChangePasswordForm()
    if form.validate_on_submit():
        if current_user.verify_password(form.old_password.data):
            current_user.password = form.password.data
            db.session.add(current_user)
            db.session.commit()
            flash('Your password has been updated.')
            return redirect(url_for('main.index'))
        else:
            flash('Invalid password.')
    return render_template("auth/change_password.html", form=form)

@auth.route('/reset', methods=['GET', 'POST'])
def password_reset_request():
    if not current_user.is_anonymous:
        return redirect(url_for('main.index'))
    form = PasswordResetRequestForm()
    if form.validate_on_submit():
        user = User.query.filter_by(email=form.email.data.lower()).fi
rst()
        if user:
            token = user.generate_reset_token()
            send_email(user.email, 'Reset Your Password',
                       'auth/email/reset_password',
                       user=user, token=token)
```

```python
        flash('An email with instructions to reset your password has been '
              'sent to you.')
        return redirect(url_for('auth.login'))
    return render_template('auth/reset_password.html', form=form)

@auth.route('/reset/<token>', methods=['GET', 'POST'])
def password_reset(token):
    if not current_user.is_anonymous:
        return redirect(url_for('main.index'))
    form = PasswordResetForm()
    if form.validate_on_submit():
        if User.reset_password(token, form.password.data):
            db.session.commit()
            flash('Your password has been updated.')
            return redirect(url_for('auth.login'))
        else:
            return redirect(url_for('main.index'))
    return render_template('auth/reset_password.html', form=form)

@auth.route('/change_email', methods=['GET', 'POST'])
@login_required
def change_email_request():
    form = ChangeEmailForm()
    if form.validate_on_submit():
        if current_user.verify_password(form.password.data):
            new_email = form.email.data.lower()
            token = current_user.generate_email_change_token(new_email)
            send_email(new_email, 'Confirm your email address',
                       'auth/email/change_email',
                       user=current_user, token=token)
            flash('An email with instructions to confirm your new email '
                  'address has been sent to you.')
            return redirect(url_for('main.index'))
        else:
            flash('Invalid email or password.')
    return render_template("auth/change_email.html", form=form)

@auth.route('/change_email/<token>')
@login_required
def change_email(token):
    if current_user.change_email(token):
        db.session.commit()
        flash('Your email address has been updated.')
    else:
        flash('Invalid request.')
    return redirect(url_for('main.index'))

# app/auth/forms.py

from flask_wtf import FlaskForm
from wtforms import StringField, PasswordField, BooleanField, SubmitField, SelectField
from wtforms.validators import DataRequired, Length, Email, Regexp, EqualTo
from wtforms import ValidationError
from ..models import User

class LoginForm(FlaskForm):
    email = StringField('Email', validators=[DataRequired(), Length(1, 64),
                                             Email()])
    password = PasswordField('Password', validators=[DataRequired()])
    remember_me = BooleanField('Keep me logged in')
    submit = SubmitField('Log In')

class RegistrationForm(FlaskForm):
    email = StringField('Email', validators=[DataRequired(), Length(1, 64),
                                             Email()])
    username = StringField('Username', validators=[
        DataRequired(), Length(1, 64),
        Regexp('^[A-Za-z][A-Za-z0-9_.]*$', 0,
               'Usernames must have only letters, numbers, dots or '
               'underscores')])
    '''role = SelectField('Role', choices=[('Customer', 'Customer'),('Employee', 'Employee'),('Manager', 'Manager')])'''
    password = PasswordField('Password', validators=[
        DataRequired(), EqualTo('password2', message='Passwords must match.')])
    password2 = PasswordField('Confirm password', validators=[DataRequired()])
    submit = SubmitField('Register')

    def validate_email(self, field):
        if User.query.filter_by(email=field.data.lower()).first():
            raise ValidationError('Email already registered.')

    def validate_username(self, field):
        if User.query.filter_by(username=field.data).first():
            raise ValidationError('Username already in use.')
```

```python

class ChangePasswordForm(FlaskForm):
    old_password = PasswordField('Old password', validators=[DataRequired()])
    password = PasswordField('New password', validators=[
        DataRequired(), EqualTo('password2', message='Passwords must match.')])
    password2 = PasswordField('Confirm new password',
                              validators=[DataRequired()])
    submit = SubmitField('Update Password')

class PasswordResetRequestForm(FlaskForm):
    email = StringField('Email', validators=[DataRequired(), Length(1, 64),
                                             Email()])
    submit = SubmitField('Reset Password')

class PasswordResetForm(FlaskForm):
    password = PasswordField('New Password', validators=[
        DataRequired(), EqualTo('password2', message='Passwords must match')])
    password2 = PasswordField('Confirm password', validators=[DataRequired()])
    submit = SubmitField('Reset Password')

class ChangeEmailForm(FlaskForm):
    email = StringField('New Email', validators=[DataRequired(), Length(1, 64),
                                                 Email()])
    password = PasswordField('Password', validators=[DataRequired()])
    submit = SubmitField('Update Email Address')

    def validate_email(self, field):
        if User.query.filter_by(email=field.data.lower()).first():
            raise ValidationError('Email already registered.')

# app/main/__init__.py

from flask import Blueprint

main = Blueprint('main', __name__)

from . import views, errors
from ..models import Permission

@main.app_context_processor
```

```
def inject_permissions():
    return dict(Permission=Permission)

# app/main/error.py

from flask import render_template
from . import main


@main.app_errorhandler(403)
def forbidden(e):
    return render_template('403.html'), 403


@main.app_errorhandler(404)
def page_not_found(e):
    return render_template('404.html'), 404


@main.app_errorhandler(500)
def internal_server_error(e):
    return render_template('500.html'), 500

# app/main/views.py

import datetime


from flask import render_template, redirect, url_for, abort, flash, request, current_app
from flask_login import login_required, current_user
from . import main

from .forms import EditProfileForm, EditProfileAdminForm, MembershipForm, PoolBookingForm, \
    FitnessBookingForm, SquashBookingForm, HallBookingForm, CardForm, RefundForm, ConfigureActivityForm, \
    ConfigureTimetableForm, \
    ConfigureFacilityForm, SearchForm
from .. import db

from ..models import Role, User, Facility, Membership, Account, Activity, Time_management, Booking, Credit_card_info, \
    Membership_type, Permission
from ..decorators import admin_required, permission_required
from ..email import send_email, str_to_pdf
import qrcode
import pdfkit


@main.route('/')
```

```python
def index():
    return render_template('index.html')

@main.route('/facility')
def facility():
    facilities = Facility.query.all()
    '''for facility in facilities:
        print(facility.name)'''
    return render_template('facility.html', facilities=facilities)

@main.route('/facility_info/<int:id>')
def view_info(id):
    facility = Facility.query.filter_by(id=id).first()
    return render_template('facility_info.html', facility=facility)

@main.route('/membership')
def view_membership_type():
    memberships = Membership_type.query.all()
    for membership in memberships:
        print(membership.length)
    return render_template('membership.html', memberships=memberships)

@main.route('/my_bookings/<int:id>')
@login_required
@permission_required(Permission.DISPLAY)
def display_bookings(id):
    account = Account.query.filter_by(user_id=id).first()
    bookings = Booking.query.order_by(Booking.timestamp.desc()).filter_by(account_id=account.id).all()
    number = Booking.query.order_by(Booking.timestamp.desc()).filter_by(account_id=account.id).count()
    facilities = Facility.query.all()
    timetable_booking = []
    for booking in bookings:
        timetable = Time_management.query.filter_by(id=booking.time_id).first()
        timetable_booking.append(timetable)

    return render_template('my_booking.html', number=number, bookings=bookings, timetable_booking=timetable_booking,
                           facilities=facilities)

@main.route('/my_card/<int:id>')
@login_required
@permission_required(Permission.DISPLAY)
```

```python
def display_cards(id):
    account = Account.query.filter_by(user_id=id).first()
    cards = Credit_card_info.query.filter_by(account_id=account.id).all()
    number = Credit_card_info.query.filter_by(account_id=account.id).count()
    return render_template('my_card.html', cards=cards, number=number)

@main.route('/remove_card/<int:id>', methods=['GET', 'POST'])
@login_required
@permission_required(Permission.DISPLAY)
def remove_card(id):
    card = Credit_card_info.query.get_or_404(id)
    db.session.delete(card)
    db.session.commit()
    flash('Successfully remove a card.')
    # The function will go to all function and render the template index.heml
    return redirect(url_for('.display_cards', id=current_user.id))

@main.route('/my_membership/<int:id>')
@login_required
@permission_required(Permission.DISPLAY)
def display_membership(id):
    account = Account.query.filter_by(user_id=id).first()
    membership = Membership.query.filter_by(account_id=account.id).first()
    membership_type = Membership_type.query.filter_by(id=membership.membership_type_id).first()
    number = Membership.query.filter_by(account_id=account.id).count()
    return render_template('my_membership.html', membership=membership, membership_type=membership_type, number=number)

@main.route('/pricing_list')
def display_pricing_list():
    facilities = Facility.query.all()
    activities = Activity.query.all()
    return render_template('pricing_list.html', facilities=facilities, activities=activities)

@main.route('/timetable_all/<int:date>/<int:offset>')
def timetable_all(date, offset):
    if date == 2 and offset == 0:
        date = 0
    if date == 2:
        date = -1
    newdate = offset + date
    new_date = datetime.date.today() + datetime.timedelta(days=newdate)
    offset += date

    timetables = Time_management.query.filter_by(date=new_date).all()
    facilities = Facility.query.all()

    return render_template('timetable_all.html', offset=offset, timetables=timetables, facilities=facilities,
                           new_date=new_date)

@main.route('/timetable_facility/<int:type>')
def timetable_facility(type):
    if type == 1:
        type = type + 1
        facility = "Swimming pool"
    elif type == 2:
        type = type + 1
        facility = "Fitness room"
    elif type == 3:
        type = type + 1
        facility = "Squash courts"
    else:
        type = 1
        facility = "Sports hall"

    facility = Facility.query.filter_by(name=facility).first()
    today = datetime.date.today()
    day2 = today + datetime.timedelta(days=1)
    day3 = today + datetime.timedelta(days=2)
    day4 = today + datetime.timedelta(days=3)
    day5 = today + datetime.timedelta(days=4)
    timetable1 = Time_management.query.filter_by(date=today).all()
    timetable2 = Time_management.query.filter_by(date=day2).all()
    timetable3 = Time_management.query.filter_by(date=day3).all()
    timetable4 = Time_management.query.filter_by(date=day4).all()
    timetable5 = Time_management.query.filter_by(date=day5).all()
    return render_template('timetable_facility.html', type=type, today=today, day3=day3, day2=day2, day4=day4,
                           day5=day5, facility=facility, timetable1=timetable1, timetable2=timetable2,
                           timetable3=timetable3, timetable4=timetable4, timetable5=timetable5)


@main.route('/book_facility/<int:type>', methods=['GET', 'POST'])
@login_required
@permission_required(Permission.PAY)
```

```
def book_facility(type):
    if type == 1:
        form = PoolBookingForm()
    elif type == 2:
        form = FitnessBookingForm()
    elif type == 3:
        form = SquashBookingForm()
    else:
        form = HallBookingForm()

    if form.validate_on_submit():
        '''activity = form.activity.data
        number = form.number.data
        date = form.date.data
        start_time = form.start_time.data
        end_time = form.end_time.data
        payment = form.payment.data'''

        fac = Facility.query.filter_by(id=type).first()
        account = Account.query.filter_by(user_id=current_user.id).first()
        for booking in account.bookings:
            book_timetable = Time_management.query.filter_by(id=booking.time_id).first()
            if book_timetable.date == form.date.data:
                flash("You can only have exactly one booking each day")
                return redirect(url_for('.book_facility', type=type))

        if form.end_time.data - form.start_time.data != 1:
            flash('You can only book a 1-hour session')
            return redirect(url_for('.book_facility', type=type))

        timetables = Time_management.query.filter_by(facility_id=type).all()
        for timetable in timetables:
            if form.start_time.data == timetable.start_time and form.date.data ==timetable.date:
                time_id = timetable.id

        timetable_booking = Time_management.query.filter_by(id=time_id).first()
        if form.number.data + timetable_booking.current_capacity > fac.capacity:
            flash('Your have too many people!')
            return redirect(url_for('.book_facility', type=type))

        else:
            account = Account.query.filter_by(user_id=current_user.id).first()
            for activity in fac.activities:
                print(activity.activity_name)
                if activity.activity_name == form.activity.data:
                    fees=activity.activity_price
                    activity_id=activity.id

            booking = Booking(number=form.number.data,
                              time_id=time_id,
                              account_id=account.id,
                              activity=form.activity.data,
                              activity_id=activity_id,
                              status="Unpaid",
                              payment=form.payment.data,
                              fees=fees)
            db.session.add(booking)
            db.session.commit()

            book = Booking.query.order_by(Booking.timestamp.desc()).filter_by(account_id=account.id).first_or_404()

            timetable_booking.current_capacity = form.number.data + timetable_booking.current_capacity
            db.session.add(timetable_booking)
            db.session.commit()

            if form.payment.data == "Credit Card":
                flash('Pay for your booking.')
                return redirect(url_for('.handle_card_booking', book_id=book.id))
            elif form.payment.data == "Cash":
                flash('You will pay for your booking by cash.')
                return redirect(url_for('.index'))

    return render_template('book_facility.html', form=form)


@main.route('/handle_card_booking/<int:book_id>', methods=['GET', 'POST'])
@login_required
@permission_required(Permission.HANDLE)
def handle_card_booking(book_id):
    account = Account.query.filter_by(user_id=current_user.id).first()
    card = Credit_card_info.query.filter_by(account_id=account.id).first()
    book = Booking.query.filter_by(id=book_id).first()
    time = Time_management.query.filter_by(id=book.time_id).first()
    form = CardForm()

    if form.validate_on_submit():
         length = len(str(form.card_number.data))
         length_year = len(str(form.expire_year.data))
         length_code = len(str(form.security_code.data))

         if card is None or card.card_number != form.card_number.data:
             if length != 16:
                 flash("Your card number should have 16 numbers.")
                 return redirect(url_for(".handle_card_booking", book_id=book_id))
             if form.expire_month.data > 12 or form.expire_month.data < 1:
                 flash("Your expire month is invalid.")
                 return redirect(url_for(".handle_card_booking", book_id=book_id))
             if length_year != 4:
                 flash("Your expire year is invalid.")
                 return redirect(url_for(".handle_card_booking", book_id=book_id))
             if length_code != 3:
                 flash("Your security code is invalid.")
                 return redirect(url_for(".handle_card_booking", book_id=book_id))

             card_info = Credit_card_info(card_number=form.card_number.data,
                                          expire_month=form.expire_month.data,
                                          expire_year=form.expire_year.data,
                                          security_code=form.security_code.data,
                                          holder_name=form.holder_name.data,
                                          account_id=account.id
                                          )
             db.session.add(card_info)
             db.session.commit()
         book.status = "Paid"
         db.session.add(book)
         db.session.commit()

         act = Activity.query.filter_by(id=book.activity_id).first()
         act.weekly_income = act.weekly_income + book.fees
         act.weekly_usage = act.weekly_usage + book.number
         db.session.add(act)
         db.session.commit()

         receipt = "Receipt:\n" + "Payer:" + current_user.username + "\n" + "item:" + book.activity + "\n" + "Fees:" + str(
             book.fees) + "\n" + book.timestamp.strftime('%Y-%m-%d %H:%M:%S') + "\n"
         pdf_name = 'app/static/' + str(book.id) + '.pdf'
         str_to_pdf(receipt, pdf_name)

         img = qrcode.make(receipt)
         img.save("app/static/receipt.jpg")
         send_email(current_user.email, 'Your booking receipt',
                    'booking_receipt', book=book, time=time)

         flash('A booking receipt has been sent to you by email.')
         return redirect(url_for("main.index"))
     if card is not None:
         form.card_number.data = card.card_number
         form.expire_month.data = card.expire_month
         form.expire_year.data = card.expire_year
         form.security_code.data = card.security_code
         form.holder_name.data = card.holder_name
     return render_template('handle_card_booking.html', book=book, time=time, form=form)


 @main.route('/cancel_booking/<int:id>', methods=['GET', 'POST'])
 @login_required
 @permission_required(Permission.CANCEL)
 def cancel_booking(id):
     booking = Booking.query.filter_by(id=id).first()
     time = Time_management.query.filter_by(id=booking.time_id).first()
     if booking.status == "Unpaid":
         db.session.delete(booking)
         db.session.commit()
         return redirect(url_for('main.display_bookings', id=current_user.id))
     else:
         form = RefundForm()
         if form.validate_on_submit():
             time.current_capacity = time.current_capacity - booking.number
             db.session.add(time)
             db.session.commit()
             db.session.delete(booking)
             db.session.commit()
             activity = Activity.query.filter_by(activity_name=booking.activity).first()
             activity.weekly_usage = activity.weekly_usage - booking.number
             activity.weekly_income = activity.weekly_income - booking.fees
             db.session.add(activity)
```

```python
            db.session.commit()
            return redirect(url_for('main.display_bookings', id=current_user.id))

    return render_template('cancel_booking.html', form=form, booking=booking, time=time)

@main.route('/pay_membership/<int:type_id>', methods=['GET', 'POST'])
@login_required
def pay_membership(type_id):
    membership_type = Membership_type.query.filter_by(id=type_id).first()
    account = Account.query.filter_by(user_id=current_user.id).first()
    form = MembershipForm()
    if form.validate_on_submit():
        membership = Membership(title=form.title.data,
                                status="Unpaid",
                                firstname=form.firstname.data,
                                lastname=form.lastname.data,
                                account_id=account.id,
                                membership_type_id=membership_type.id
                                )
        db.session.add(membership)
        db.session.commit()

        membership = Membership.query.filter_by(account_id=account.id).first()
        if (membership_type.length == 1):
            length = 30
        elif (membership_type.length == 3):
            length = 90
        else:
            length = 365
        membership.valueOfEnd_date(length)
        db.session.commit()

        if form.payment.data == "Credit Card":
            flash('Pay for your booking.')
            return redirect(url_for('.handle_card_membership', id=account.id))
        elif form.payment.data == "Cash":
            flash('You will pay for your booking by cash.')
            return redirect(url_for('.index'))

    return render_template('pay_membership.html', form=form, membership_type=membership_type)
```

```python

@main.route('/handle_card_membership/<int:id>', methods=['GET', 'POST'])
@login_required
def handle_card_membership(id):
    membership = Membership.query.filter_by(account_id=id).first()
    card = Credit_card_info.query.filter_by(account_id=id).first()
    form = CardForm()
    membership_type = Membership_type.query.filter_by(id=membership.membership_type_id).first()
    price = membership_type.price
    if form.validate_on_submit():
        card_info = Credit_card_info(card_number=form.card_number.data,
                                     expire_month=form.expire_month.data,
                                     expire_year=form.expire_year.data,
                                     security_code=form.security_code.data,
                                     holder_name=form.holder_name.data,
                                     account_id=id
                                     )
        db.session.add(card_info)
        db.session.commit()
        membership.status = "Paid"
        db.session.add(membership)
        db.session.commit()
        return redirect(url_for('main.index'))

    if card is not None:
        form.card_number.data = card.card_number
        form.expire_month.data = card.expire_month
        form.expire_year.data = card.expire_year
        form.security_code.data = card.security_code
        form.holder_name.data = card.holder_name

    return render_template('handle_card_membership.html', price=price, form=form)

@main.route('/configure_facility', methods=['GET', 'POST'])
@login_required
def configure_facility():
    form = ConfigureFacilityForm()
    if form.validate_on_submit():
        operation = form.operation.data
        capacity = form.capacity.data
        name = form.name.data
        url = form.url.data
        description = form.description.data

        if operation == "add":
            facility = Facility(name=name,
                                url=url,
                                capacity=capacity,
                                description=description)
            db.session.add(facility)
            db.session.commit()
            flash('You have added the facility')
            return redirect(url_for('main.index'))

        elif operation == "edit":
            fac = Facility.query.filter_by(name=name).first()
            fac.capacity = capacity
            fac.url = url
            fac.description = description

            db.session.add(fac)
            db.session.commit()
            flash('You have edited the facility')
            return redirect(url_for('main.index'))

        elif operation == "delete":
            fac = Facility.query.filter_by(name=name).first()
            db.session.delete(fac)
            db.session.commit()
            flash('You have deleted the facility')
            return redirect(url_for('main.index'))

    return render_template('configure_facility.html', form=form)

@main.route('/configure_activity', methods=['GET', 'POST'])
@login_required
def configure_activity():
    form = ConfigureActivityForm()
    if form.validate_on_submit():

        fac = Facility.query.filter_by(name=form.facility.data).first()
        if form.operation.data == "add":
            act = Activity(weekly_income=0,
                           weekly_usage=0,
                           activity_price=form.price.data,
                           activity_name=form.activity.data,
                           facility_id=fac.id
                           )
            db.session.add(act)
            db.session.commit()
            flash('You have added the activity')
            return redirect(url_for('main.index'))

        elif form.operation.data == "edit":
            act = Activity.query.filter_by(activity_name=form.activity.data).first()
            act.activity_price = form.price.data,
            act.activity_name = form.activity.data
            act.facility_id = fac.id

            db.session.add(act)
            db.session.commit()
            flash('You have edited the activity')
            return redirect(url_for('main.index'))

        elif form.operation.data == "delete":
            act = Activity.query.filter_by(activity_name=form.activity.data).first()
            db.session.delete(act)
            db.session.commit()
            flash('You have deleted the activity')
            return redirect(url_for('main.index'))

    return render_template('configure_activity.html', form=form)


@main.route('/configure_timetable', methods=['GET', 'POST'])
@login_required
def configure_timetable():
    form = ConfigureTimetableForm()
    if form.validate_on_submit():
        fac = Facility.query.filter_by(name=form.facility.data).first()

        if fac is None:
            flash("The facility is not existed")
            return redirect(url_for("main.configure_timetable"))

        for i in range(form.start_time.data, form.end_time.data):
            timetable = Time_management(date=form.date.data,
                                        start_time=i,
                                        end_time=i + 1,
                                        facility_id=fac.id)
            db.session.add(timetable)
            db.session.commit()
        flash("Sucessful operation!")
        return redirect(url_for('main.index'))
    return render_template('configure_timetable.html', form=form)
```

```python

@main.route('/cancel_membership/<int:id>', methods=['GET', 'POST'])
@login_required
@permission_required(Permission.CANCEL)
def cancel_membership(id):
    membership = Membership.query.filter_by(id=id).first()
    timedelta = membership.end_date - datetime.datetime.utcnow()
    days_left = timedelta.days
    money_refund = timedelta.days * 0.5
    form = RefundForm()
    if form.validate_on_submit():
        db.session.delete(membership)
        db.session.commit()
        flash('You have cancelled our membership!')
        return redirect(url_for('main.display_membership', id=current_user.id))
    return render_template('cancel_membership.html', money_refund=money_refund, days_left=days_left,
                           membership=membership, form=form)

@main.route('/user/<username>')
def user(username):
    user = User.query.filter_by(username=username).first_or_404()
    return render_template('user.html', user=user)

@main.route('/search_booking', methods=['GET', 'POST'])
@login_required
@permission_required(Permission.CANCEL)
def search_booking():
    form = SearchForm()
    if form.validate_on_submit():
        email = form.email.data

        user = User.query.filter_by(email=email).first()
        return redirect(url_for('main.display_bookings', id=user.id))
    return render_template('search_booking.html', form=form)

@main.route('/search_membership', methods=['GET', 'POST'])
@login_required
@permission_required(Permission.CANCEL)
def search_membership():
    form = SearchForm()
    if form.validate_on_submit():
        email = form.email.data

        user = User.query.filter_by(email=email).first()
        return redirect(url_for('main.display_membership', id=user.id))
    return render_template('search_membership.html', form=form)


@main.route('/view_income')
@login_required
def view_income():
    session1_income, session2_income, session3_income, session4_income, session5_income, session6_income, session14_income = 0, 0, 0, 0, 0, 0, 0
    session7_income, session8_income, session9_income, session10_income, session11_income, session12_income, session13_income = 0, 0, 0, 0, 0, 0, 0
    session1_usage, session2_usage, session3_usage, session4_usage, session5_usage, session6_usage, session14_usage = 0, 0, 0, 0, 0, 0, 0
    session7_usage, session8_usage, session9_usage, session10_usage, session11_usage, session12_usage, session13_usage = 0, 0, 0, 0, 0, 0, 0
    time8 = Time_management.query.filter_by(start_time=8).all()
    for timetable in time8:
        for booking in timetable.bookings:
            session1_income += booking.fees
            session1_usage += booking.number

    time9 = Time_management.query.filter_by(start_time=9).all()
    for timetable in time9:
        for booking in timetable.bookings:
            session2_income += booking.fees
            session2_usage += booking.number

    time10 = Time_management.query.filter_by(start_time=10).all()
    for timetable in time10:
        for booking in timetable.bookings:
            session3_income += booking.fees
            session3_usage += booking.number

    time11 = Time_management.query.filter_by(start_time=11).all()
    for timetable in time11:
        for booking in timetable.bookings:
            session4_income += booking.fees
            session4_usage += booking.number

    time12 = Time_management.query.filter_by(start_time=12).all()
    for timetable in time12:
        for booking in timetable.bookings:
            session5_income += booking.fees
            session5_usage += booking.number

    time13 = Time_management.query.filter_by(start_time=13).all()
    for timetable in time13:
        for booking in timetable.bookings:
            session6_income += booking.fees
            session6_usage += booking.number

    time14 = Time_management.query.filter_by(start_time=14).all()
    for timetable in time14:
        for booking in timetable.bookings:
            session7_income += booking.fees
            session7_usage += booking.number

    time15 = Time_management.query.filter_by(start_time=15).all()
    for timetable in time15:
        for booking in timetable.bookings:
            session8_income += booking.fees
            session8_usage += booking.number

    time16 = Time_management.query.filter_by(start_time=16).all()
    for timetable in time16:
        for booking in timetable.bookings:
            session9_income += booking.fees
            session9_usage += booking.number

    time17 = Time_management.query.filter_by(start_time=17).all()
    for timetable in time17:
        for booking in timetable.bookings:
            session10_income += booking.fees
            session10_usage += booking.number

    time18 = Time_management.query.filter_by(start_time=18).all()
    for timetable in time18:
        for booking in timetable.bookings:
            session11_income += booking.fees
            session11_usage += booking.number

    time19 = Time_management.query.filter_by(start_time=19).all()
    for timetable in time19:
        for booking in timetable.bookings:
            session12_income += booking.fees
            session12_usage += booking.number

    time20 = Time_management.query.filter_by(start_time=20).all()
    for timetable in time20:
        for booking in timetable.bookings:
            session13_income += booking.fees
            session13_usage += booking.number

    time21 = Time_management.query.filter_by(start_time=21).all()
    for timetable in time21:
        for booking in timetable.bookings:
            session14_income += booking.fees
            session14_usage += booking.number

    overall_income = 0
    overall_usage = 0
    activities = Activity.query.all()
    for activity in activities:
        overall_income += activity.weekly_income
        overall_usage += activity.weekly_usage
    return render_template('business.html', session14_usage=session14_usage, session14_income=session14_income,
                           session13_usage=session13_usage, session13_income=session13_income,
                           session12_usage=session12_usage, session12_income=session12_income,
                           session11_usage=session11_usage, session11_income=session11_income,
                           session10_usage=session10_usage, session10_income=session10_income,
                           session9_usage=session9_usage, session9_income=session9_income,
                           session8_usage=session8_usage, session8_income=session8_income,
                           session7_usage=session7_usage, session7_income=session7_income,
                           session6_usage=session6_usage,
                           session6_income=session6_income, session5_usage=session5_usage,
                           session5_income=session5_income, session4_usage=session4_usage,
                           session4_income=session4_income,
                           session3_usage=session3_usage, session3_income=session3_income,
                           session2_usage=session2_usage, session2_income=session2_income,
                           session1_usage=session1_usage, session1_income=session1_income,
                           overall_income=overall_income, overall_usage=overall_usage, activities=activities)


@main.route('/edit-profile', methods=['GET', 'POST'])
@login_required
def edit_profile():
    form = EditProfileForm()
    if form.validate_on_submit():
        current_user.username = form.username.data
        db.session.add(current_user._get_current_object())
        db.session.commit()
        flash('Your profile has been updated.')
        return redirect(url_for('.user', username=current_user.username))
    form.name.data = current_user.name
    return render_template('edit_profile.html', form=form)


@main.route('/edit-profile/<int:id>', methods=['GET', 'POST'])
@login_required
@admin_required
def edit_profile_admin(id):
    user = User.query.get_or_404(id)
    form = EditProfileAdminForm(user=user)
    if form.validate_on_submit():
        user.email = form.email.data
        user.username = form.username.data
        user.confirmed = form.confirmed.data
        user.role = Role.query.get(form.role.data)
        db.session.add(user)
        db.session.commit()
        flash('The profile has been updated.')
        return redirect(url_for('.user', username=user.username))
    form.email.data = user.email
    form.username.data = user.username
    form.confirmed.data = user.confirmed
    form.role.data = user.role_id
    return render_template('edit_profile.html', form=form, user=user)

# app/main/forms.py

from flask_wtf import FlaskForm

from wtforms import StringField, TextAreaField, BooleanField, SelectField,\
    SubmitField,IntegerField,RadioField,DateField
from wtforms.validators import DataRequired, Length, Email, Regexp
from wtforms import ValidationError
from ..models import Role, User


class MembershipForm(FlaskForm):
    title = SelectField('Title', choices=[('Ms', 'Ms'), ('Mrs', 'Mrs'), ('Mr', 'Mr')], validators=[DataRequired()])
    firstname = StringField('First name', validators=[
        DataRequired(), Length(1, 64), ])
    lastname = StringField('Last name', validators=[
        DataRequired(), Length(1, 64), ])
    streetname = StringField('House number and Street name', validators=[DataRequired()])
    city = StringField('Town/City', validators=[DataRequired()])
    county = StringField('County', validators=[DataRequired()])
    postcode = StringField('Postcode', validators=[DataRequired()])
    phone = IntegerField('Area code + Phone number', validators=[DataRequired()])
    payment = RadioField('Payment method', choices=[('Credit Card', 'Credit Card'),('Cash', 'Cash')],
                         validators=[DataRequired()])
    submit = SubmitField('Submit')

class RefundForm(FlaskForm):
    card_number = StringField('Card number', validators=[Length(0, 64)])
    expire_month = IntegerField('Expire month', validators=[DataRequired()])
    expire_year = IntegerField('Expire year', validators=[DataRequired()])
    security_code = IntegerField('Security code', validators=[DataRequired()])
    submit = SubmitField('Cancel it now', validators=[DataRequired()])

class BasicBookingForm(FlaskForm):
    number = IntegerField('Numbers of people:')
    date = DateField('Date(e.g 2020-05-13):', validators=[DataRequired()],format='%Y-%m-%d')
    start_time = IntegerField('Start time(e.g 19):', validators=[DataRequired()])
    end_time = IntegerField('End time(e.g 20):', validators=[DataRequired()])
    payment = RadioField('Payment method', choices=[('Credit Card', 'Credit Card'),('Cash', 'Cash')],
                         validators=[DataRequired()])


class PoolBookingForm(BasicBookingForm):
    activity = SelectField('Activity',choices=[('General use','General use'),('Lane swimming','Lane swimming'),
                                               ('Lessons','Lessons'),('Team events','Team events')])
    submit = SubmitField('Book')

class FitnessBookingForm(BasicBookingForm):
    activity = SelectField('Activity', choices=[('General use', 'General use')])
    submit = SubmitField('Book')

class SquashBookingForm(BasicBookingForm):
    activity = SelectField('Activity', choices=[('General use', 'General use'),('Team events','Team events')])
```

```python
    submit = SubmitField('Book')

class HallBookingForm(BasicBookingForm):
    activity = SelectField('Activity', choices=[('1-hour sessions', '1-hour sessions')])
    submit = SubmitField('Book')

class EditProfileForm(FlaskForm):
    name = StringField('Real name', validators=[Length(0, 64)])
    location = StringField('Location', validators=[Length(0, 64)])
    about_me = TextAreaField('About me')
    submit = SubmitField('Submit')

class CardForm(FlaskForm):
    card_number = IntegerField('Card number',validators=[DataRequired()])
    holder_name = StringField('Name on account:', validators=[Length(0, 64)])
    expire_month = IntegerField('Expire month',validators=[DataRequired()])
    expire_year = IntegerField('Expire year', validators=[DataRequired()])
    security_code = IntegerField('Security code', validators=[DataRequired()])
    submit = SubmitField('Pay securely now', validators=[DataRequired()])

class ConfigureFacilityForm(FlaskForm):
    operation = SelectField('Operation', choices=[('add', 'add'), ('edit', 'edit'),('delete', 'delete')])
    name = StringField('Facility name',validators=[Length(0, 64)])
    capacity = IntegerField('Capacity',validators=[DataRequired()])
    url = StringField('Photo url',validators=[Length(0, 64)])
    description = StringField('Description',validators=[Length(0, 64)])
    submit = SubmitField('Submit', validators=[DataRequired()])

class ConfigureActivityForm(FlaskForm):
    operation = SelectField('Operation', choices=[('add', 'add'), ('edit', 'edit'),('delete', 'delete')])
    facility = StringField('Facility name',validators=[Length(0, 64)])
    price = IntegerField('Price',validators=[DataRequired()])
    activity = StringField('Activity name',validators=[Length(0, 64)])
    submit = SubmitField('Submit', validators=[DataRequired()])

class ConfigureTimetableForm(FlaskForm):
    facility = StringField('Facility name', validators=[Length(0, 64)])
    date = DateField('Date',validators=[DataRequired()])
    start_time = IntegerField('Open time',validators=[DataRequired()])
    end_time = IntegerField('Close time',validators=[DataRequired()])
    submit = SubmitField('Submit', validators=[DataRequired()])

class SearchForm(FlaskForm):
    email = StringField('Email', validators=[DataRequired(), Length(1, 64),
                                             Email()])
    submit = SubmitField('Submit', validators=[DataRequired()])

class EditProfileAdminForm(FlaskForm):
    email = StringField('Email', validators=[DataRequired(), Length(1, 64),
                                             Email()])
    username = StringField('Username', validators=[
        DataRequired(), Length(1, 64),
        Regexp('^[A-Za-z][A-Za-z0-9_.]*$', 0,
               'Usernames must have only letters, numbers, dots or '
               'underscores')])
    confirmed = BooleanField('Confirmed')
    role = SelectField('Role', coerce=int)
    name = StringField('Real name', validators=[Length(0, 64)])
    location = StringField('Location', validators=[Length(0, 64)])
    about_me = TextAreaField('About me')
    submit = SubmitField('Submit')

    def __init__(self, user, *args, **kwargs):
        super(EditProfileAdminForm, self).__init__(*args, **kwargs)
        self.role.choices = [(role.id, role.name)
                             for role in Role.query.order_by(Role.name).all()]
        self.user = user

    def validate_email(self, field):
        if field.data != self.user.email and \
                User.query.filter_by(email=field.data).first():
            raise ValidationError('Email already registered.')

    def validate_username(self, field):
        if field.data != self.user.username and \
                User.query.filter_by(username=field.data).first():
            raise ValidationError('Username already in use.')

```

```css
/* app/static/style.css */

.profile-thumbnail {
    position: absolute;
}
.profile-header {
    min-height: 260px;
    margin-left: 280px;
}

<!-- app/templates/403.html -->

{% extends "base.html" %}

{% block title %}Flasky - Forbidden{% endblock %}

{% block page_content %}
<div class="page-header">
    <h1>Forbidden</h1>
</div>
{% endblock %}

<!-- app/templates/404.html -->
{% extends "base.html" %}

{% block title %}Flasky - Page Not Found{% endblock %}

{% block page_content %}
<div class="page-header">
    <h1>Not Found</h1>
</div>
{% endblock %}

<!-- app/templates/500.html -->
{% extends "base.html" %}

{% block title %}Flasky - Internal Server Error{% endblock %}

{% block page_content %}
<div class="page-header">
    <h1>Internal Server Error</h1>
</div>
{% endblock %}

<!-- app/templates/base.html -->
{% extends "bootstrap/base.html" %}

{% block title %}Flasky{% endblock %}

{% block head %}
{{ super() }}
```

```
<link rel="shortcut icon" href="{{ url_for('static', filename='favicon.ico') }}" type="image/x-icon">
<link rel="icon" href="{{ url_for('static', filename='favicon.ico') }}" type="image/x-icon">
<link rel="stylesheet" type="text/css" href="{{ url_for('static', filename='styles.css') }}">
{% endblock %}

{% block navbar %}
<div class="navbar navbar-inverse" role="navigation">
    <div class="container">
        <div class="navbar-header">
            <button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse">
                <span class="sr-only">Toggle navigation</span>
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
            </button>
            <a class="navbar-brand" href="{{ url_for('main.index')}}">The Gym</a>
        </div>
        <div class="navbar-collapse collapse">
            <ul class="nav navbar-nav">

                {% if current_user.role_id == 1 %}
                    <li><a href="{{ url_for('main.facility')}}">Facility</a></li>
                    <li><a href="{{ url_for('main.display_pricing_list')}}">Pricing List</a></li>
                    <li><a href="{{ url_for('main.view_membership_type')}}">Membership</a></li>
                    <li class="dropdown">
                        <a href="#" class="dropdown-toggle" data-toggle="dropdown">
                            Timetable <b class="caret"></b>
                        </a>
                        <ul class="dropdown-menu">
                            <li><a href="{{ url_for('main.timetable_all', date=0, offset=0) }}">All Timetables</a></li>
                            <li><a href="{{ url_for('main.timetable_facility',type=1) }}">One specified facility</a></li>
                        </ul>
                    </li>

                {% elif current_user.role_id == 2 %}
```

```
                        <li><a href="{{ url_for('main.facility')}}">Facility</a></li>
                        <li><a href="{{ url_for('main.display_pricing_list')}}">Pricing List</a></li>
                        <li><a href="{{ url_for('main.search_booking') }}">Search Booking</a></li>
                        <li><a href="{{ url_for('main.search_booking') }}">Search Membership</a></li>
                        <li class="dropdown">
                            <a href="#" class="dropdown-toggle" data-toggle="dropdown">
                                Timetable <b class="caret"></b>
                            </a>
                            <ul class="dropdown-menu">
                                <li><a href="{{ url_for('main.timetable_all', date=0, offset=0) }}">All Timetables</a></li>
                                <li><a href="{{ url_for('main.timetable_facility',type=1) }}">One specified facility</a></li>
                            </ul>
                        </li>

                    {% elif current_user.role_id == 3 %}
                        <li><a href="{{ url_for('main.facility')}}">Facility</a></li>
                            <li><a href="{{ url_for('main.display_pricing_list')}}">Pricing List</a></li>
                            <li><a href="{{ url_for('main.view_income') }}">Business</a></li>
                            <li class="dropdown">
                                <a href="#" class="dropdown-toggle" data-toggle="dropdown">
                                    Configure <b class="caret"></b>
                                </a>
                                <ul class="dropdown-menu">
                                    <li><a href="{{ url_for('main.configure_facility') }}">Facility</a></li>
                                    <li><a href="{{ url_for('main.configure_activity') }}">Activity</a></li>
                                    <li><a href="{{ url_for('main.configure_timetable') }}">Timetable</a></li>
                                </ul>
                            </li>
                            <li class="dropdown">
                                <a href="#" class="dropdown-toggle" data-toggle="dropdown">
                                    Timetable <b class="caret"></b>
                                </a>
                                <ul class="dropdown-menu">
                                    <li><a href="{{ url_for('main.timetable_all', date=0, offset=0) }}">All Timetables</a></li>
                                    <li><a href="{{ url_for('main.timetable_facility',type=1) }}">One specified facility</a></li>
                                </ul>
                            </li>
                    {% else %}
                           <li><a href="{{ url_for('main.facility')}}">Facility</a></li>
                        <li><a href="{{ url_for('main.display_pricing_list')}}">Pricing List</a></li>
                        <li><a href="{{ url_for('main.view_membership_type')}}">Membership</a></li>
                        <li class="dropdown">
                            <a href="#" class="dropdown-toggle" data-toggle="dropdown">
                                Timetable <b class="caret"></b>
                            </a>
                            <ul class="dropdown-menu">
                                <li><a href="{{ url_for('main.timetable_all', date=0, offset=0) }}">All Timetables</a></li>
                                <li><a href="{{ url_for('main.timetable_facility',type=1) }}">One specified facility</a></li>
                            </ul>
                        </li>
                    {% endif %}

                </ul>

                <ul class="nav navbar-nav navbar-right">
                    {% if current_user.is_authenticated %}
                    <li class="dropdown">
                        <a href="#" class="dropdown-toggle" data-toggle="dropdown">
                            <img src="{{ current_user.gravatar(size=18) }}">
                            Account <b class="caret"></b>
                        </a>
                        <ul class="dropdown-menu">
                            <li class="page-header">
                            <h3>Hello, {% if current_user.is_authenticated %}{{ current_user.username }}{% else %}Stranger{% endif %}!</h3>
                            </li>
                        {% if current_user.role_id == 1 %}
                            <li><a href="{{ url_for('main.display_bookings',id=current_user.id) }}">My bookings</a></li>
                            <li><a href="{{ url_for('main.display_cards',id=current_user.id) }}">My Card</a></li>
                            <li><a href="{{ url_for('main.display_membership',id=current_user.id) }}">My Membership</a></li>
                        {% endif %}
```

```
                        <li><a href="{{ url_for('auth.change_password') }}">Change Password</a></li>
                        <li><a href="{{ url_for('auth.change_email_request') }}">Change Email</a></li>
                        <li><a href="{{ url_for('auth.logout') }}">Log Out</a></li>
                        {% if current_user.is_authenticated %}
                            <li><a href="{{ url_for('main.edit_profile') }}">Change Profile</a></li>
                        {% endif %}
                    </ul>
                </li>
                {% else %}
                <li><a href="{{ url_for('auth.login') }}">Log In</a></li>
                {% endif %}
            </ul>
        </div>
    </div>
</div>
{% endblock %}

{% block content %}
<div class="container">
    {% for message in get_flashed_messages() %}
    <div class="alert alert-warning">
        <button type="button" class="close" data-dismiss="alert">&times;</button>
        {{ message }}
    </div>
    {% endfor %}

    {% block page_content %}{% endblock %}
</div>
{% endblock %}

{% block scripts %}
{{ super() }}
{{ moment.include_moment() }}
{% endblock %}

<!-- app/templates/book_facility.html -->
{% extends "base.html" %}
{% import "bootstrap/wtf.html" as wtf %}

{% block title %}Book Facility{% endblock %}

{% block page_content %}
    <div class="row">
        <div class="col-md-6 col-xs-9">
```

```
            <h3>Book Facility:</h3>
                {{ wtf.quick_form(form) }}
            </div>
        </div>
{% endblock %}

<!-- app/templates/booking_receipt.html -->
<p>Dear {{ current_user.username }},</p>
<p>Your booking receipt:</p>
<p>*********************************</p>
<p> Activity: {{ book.activity }}</p>
<p> Date:{{ time.date }}</p>
<p> Time:{{ time.start_time }}:00 ~ {{ time.end_time }}:00</p>
<p>*********************************</p>
<p>Total: {{book.fees}}</p>
</br>
</br>
</br>
<p>{{ book.timestamp }}</p>
<p><img src="../static/receipt.jpg"></p>

<!-- app/templates/business.html -->
{% extends "base.html" %}
{% import "bootstrap/wtf.html" as wtf %}


{% block title %}Timetable all{% endblock %}

{% block page_content %}

<div class="container-fluid">
            <div class="col-md-10 col-md-offset-1 col-xs-10 col-md-
offset-1">

            <h3 id="cart">Overall weekly business: </h3>
            <h4>Overall income:{{ overall_income }}</h4>
            <h4>Overall usage:{{ overall_usage }}</h4>
    <br>
            <h3 id="cart">Per-activity weekly business: </h3>
                <table class="table table-striped table-
hover mytable" id="complete-
table" width='100%' border='0' cellspacing='0' cellpadding='0'  style='ta
ble-layout: fixed'>
                    <tbody>
                        <tr>
                            <th colspan='3' style='word-wrap: break-
word'>Activity</th>
                            <th colspan='3' style='word-wrap: break-
word'>Income</th>
```

```
                            <th colspan='3' style='word-wrap: break-word'>Usage </th>
                        </tr>

                         {% for activity in activities %}
                             <tr>
                                <td colspan='3' style='word-wrap: break-word'>{{ activity.activity_name}}</td>
                                <td colspan='3' style='word-wrap: break-word'>{{ activity.weekly_income }}</td>
                                <td colspan='3' style='word-wrap: break-word'>{{ activity.weekly_usage }}</td>
                            </tr>
                         {% endfor %}
                    </tbody>
                </table>

            <h3 id="cart">Per-session weekly business: </h3>
                <table class="table table-striped table-hover mytable" id="complete-table" width='100%' border='0' cellspacing='0' cellpadding='0'  style='table-layout: fixed'>
                    <tbody>
                        <tr>
                            <th colspan='3' style='word-wrap: break-word'>Session</th>
                            <th colspan='3' style='word-wrap: break-word'>Income</th>
                            <th colspan='3' style='word-wrap: break-word'>Usage </th>
                        </tr>

                        <tr>
                            <td colspan='3' style='word-wrap: break-word'>8:00 ~ 9:00</td>
                            <td colspan='3' style='word-wrap: break-word'>{{ session1_income }}</td>
                            <td colspan='3' style='word-wrap: break-word'>{{ session1_usage }}</td>
                        </tr>

                        <tr>
                            <td colspan='3' style='word-wrap: break-word'>9:00 ~ 10:00</td>
                            <td colspan='3' style='word-wrap: break-word'>{{ session2_income }}</td>
                            <td colspan='3' style='word-wrap: break-word'>{{ session2_usage }}</td>
                        </tr>

                    <tr>
                            <td colspan='3' style='word-wrap: break-word'>10:00 ~ 11:00</td>
                            <td colspan='3' style='word-wrap: break-word'>{{ session3_income }}</td>
                            <td colspan='3' style='word-wrap: break-word'>{{ session3_usage }}</td>
                        </tr>

                    <tr>
                            <td colspan='3' style='word-wrap: break-word'>11:00 ~ 12:00</td>
                            <td colspan='3' style='word-wrap: break-word'>{{ session4_income }}</td>
                            <td colspan='3' style='word-wrap: break-word'>{{ session4_usage }}</td>
                        </tr>

                    <tr>
                            <td colspan='3' style='word-wrap: break-word'>12:00 ~ 13:00</td>
                            <td colspan='3' style='word-wrap: break-word'>{{ session5_income }}</td>
                            <td colspan='3' style='word-wrap: break-word'>{{ session5_usage }}</td>
                        </tr>

                        <tr>
                            <td colspan='3' style='word-wrap: break-word'>13:00 ~ 14:00</td>
                            <td colspan='3' style='word-wrap: break-word'>{{ session6_income }}</td>
                            <td colspan='3' style='word-wrap: break-word'>{{ session6_usage }}</td>
                        </tr>
                        <tr>
                            <td colspan='3' style='word-wrap: break-word'>14:00 ~ 15:00</td>
                            <td colspan='3' style='word-wrap: break-word'>{{ session7_income }}</td>
                            <td colspan='3' style='word-wrap: break-word'>{{ session7_usage }}</td>
                        </tr>
                        <tr>
                            <td colspan='3' style='word-wrap: break-word'>15:00 ~ 16:00</td>
                            <td colspan='3' style='word-wrap: break-word'>{{ session8_income }}</td>
                            <td colspan='3' style='word-wrap: break-word'>{{ session8_usage }}</td>
                        </tr>
                        <tr>
                            <td colspan='3' style='word-wrap: break-word'>16:00 ~ 17:00</td>
                            <td colspan='3' style='word-wrap: break-word'>{{ session9_income }}</td>
                            <td colspan='3' style='word-wrap: break-word'>{{ session9_usage }}</td>
                        </tr>
                        <tr>
                            <td colspan='3' style='word-wrap: break-word'>17:00 ~ 18:00</td>
                            <td colspan='3' style='word-wrap: break-word'>{{ session10_income }}</td>
                            <td colspan='3' style='word-wrap: break-word'>{{ session10_usage }}</td>
                        </tr>

                    <tr>
                            <td colspan='3' style='word-wrap: break-word'>18:00 ~ 19:00</td>
                            <td colspan='3' style='word-wrap: break-word'>{{ session11_income }}</td>
                            <td colspan='3' style='word-wrap: break-word'>{{ session11_usage }}</td>
                        </tr>

                    <tr>
                            <td colspan='3' style='word-wrap: break-word'>19:00 ~ 20:00</td>
                            <td colspan='3' style='word-wrap: break-word'>{{ session12_income }}</td>
                            <td colspan='3' style='word-wrap: break-word'>{{ session12_usage }}</td>
                        </tr>
                    <tr>
                            <td colspan='3' style='word-wrap: break-word'>20:00 ~ 21:00</td>
                            <td colspan='3' style='word-wrap: break-word'>{{ session13_income }}</td>
                            <td colspan='3' style='word-wrap: break-word'>{{ session13_usage }}</td>
                        </tr>
                     <tr>
                            <td colspan='3' style='word-wrap: break-word'>21:00 ~ 22:00</td>
                            <td colspan='3' style='word-wrap: break-word'>{{ session14_income }}</td>
                            <td colspan='3' style='word-wrap: break-word'>{{ session14_usage }}</td>
```

```
                            </tr>

                        </tbody>
                    </table>
                </div>
</div>
{% endblock %}

<!-- app/templates/cancel_booking.html -->
{% extends "base.html" %}
{% import "bootstrap/wtf.html" as wtf %}

{% block title %}Cancel Membership{% endblock %}

{% block page_content %}
    <div class="row">
        <div class="col-md-6 col-xs-9">
            <h3>Booking information:</h3>
            <h4>Number: {{ booking.number }}</h4>
            <h4>Activity: {{ booking.activity }}</h4>
            <h4>Date: {{ time.date }}</h4>
            <h4>Time: {{ time.start_time }}:00 ~ {{ time.end_time }}
:00</h4>

            <h3>Refund: ${{ booking.fees }}</h3>

            <h3>Refund information:</h3>
                    {{ wtf.quick_form(form) }}
            </div>
        </div>
{% endblock %}

<!-- app/templates/cancel_membership.html -->
{% extends "base.html" %}
{% import "bootstrap/wtf.html" as wtf %}

{% block title %}Cancel Membership{% endblock %}

{% block page_content %}
    <div class="row">
        <div class="col-md-6 col-xs-9">
            <h3>Membership information:</h3>
            <h4>Status: {{ membership.status }}</h4>
```

```
                <h4>Start date: {{ membership.start_date }}</h4>
                <h4>End date: {{ membership.end_date}}</h4>
                <h4>Days left: {{ days_left}}</h4>

                <h3>Refund information:</h3>
                    {{ wtf.quick_form(form) }}
                </div>
            </div>
            <div class="col-md-4 col-xs-6">
                <h3>Refund: ${{ money_refund }}</h3>
            </div>
        </div>

{% endblock %}

<!-- app/templates/configure_activity.html -->
{% extends "base.html" %}
{% import "bootstrap/wtf.html" as wtf %}

{% block title %}Configure Activity{% endblock %}

{% block page_content %}
        <div class="row">
            <div class="col-md-6 col-xs-9">
                <h3>Configure Activity:</h3>
                        {{ wtf.quick_form(form) }}
                </div>
            </div>
{% endblock %}

<!-- app/templates/configure_facility.html -->
{% extends "base.html" %}
{% import "bootstrap/wtf.html" as wtf %}

{% block title %}Configure Facility{% endblock %}

{% block page_content %}
        <div class="row">
            <div class="col-md-6 col-xs-9">
                <h3>Configure Facility:</h3>

                        {{ wtf.quick_form(form) }}

                </div>
            </div>
{% endblock %}

<!-- app/templates/configure_timetable.html -->
```

```
{% extends "base.html" %}
{% import "bootstrap/wtf.html" as wtf %}

{% block title %}Configure Timetable{% endblock %}

{% block page_content %}
    <div class="row">
        <div class="col-md-6 col-xs-9">
            <h3>Configure Timetable:</h3>

                    {{ wtf.quick_form(form) }}

            </div>
        </div>
{% endblock %}

<!-- app/templates/edit_profile.html -->
{% extends "base.html" %}
{% import "bootstrap/wtf.html" as wtf %}

{% block title %}Flasky - Edit Profile{% endblock %}

{% block page_content %}
<div class="page-header">
    <h1>Edit Your Profile</h1>
</div>
<div class="col-md-4">
    {{ wtf.quick_form(form) }}
</div>
{% endblock %}

<!-- app/templates/facility.html -->
{% extends "base.html" %}

{% block title %}Facility{% endblock %}

{% block page_content %}
    <div class="container-fluid">
    {%  for facility in facilities %}
            <div class="col-md-4 col-xs-6 ">
                <a href="{{ url_for('.view_info', id=facility.id)}}"
>
                    <img src={{ facility.url }} class="img-
responsive" alt="Responsive image" />
                </a>
                <a href="{{ url_for('.view_info',id=facility.id)}}">
                    <h4>{{ facility.name }}</h4>
                </a>
```

```
                    <a href="{{ url_for('.view_info',id=facility.id)}}">
                        <h5>View more</h5>
                    </a>
                </div>
        {% endfor %}
</div>
{% endblock %}

<!-- app/templates/facility_info.html -->
{% extends "base.html" %}

{% block title %}Facility Detail{% endblock %}

{% block page_content %}
<div class="good-form">
    <div class="row">
        <div class="col-md-5 col-md-offset-1 col-xs-7 col-md-offset-2">
            <img src={{ facility.url }} class="img-responsive" alt="Responsive image"/>
        </div>
        <div class="col-md-5 col-xs-6">
            <h3>Name:</br>{{ facility.name }}</h3>
            <h3>Description:</br>{{ facility.description }}</h3>
            <h3>Capacity:</br>{{ facility.capacity }}</h3>
            <h3>Activity:
            {% for activity in facility.activities %}
                </br>
                {{ activity.activity_name }}
            {% endfor %}
            </h3>
        </div>
    </div>
</div>

{% endblock %}

<!-- app/templates/handle_card_booking.html -->
{% extends "base.html" %}
{% import "bootstrap/wtf.html" as wtf %}


{% block title %}Membership{% endblock %}

{% block page_content %}
    <div class="row">
        <div class="col-md-6 col-xs-9">
            <h3>Booking information:</h3>
            <h4>Number: {{ book.number }}</h4>
            <h4>Activity: {{ book.activity }}</h4>
            <h4>Date: {{ time.date }}</h4>
            <h4>Time: {{time.start_time }}:00 ~ {{ time.end_time }}:00</h4>
            <h4>Total: ${{ book.fees }}</h4>

            <h3>Payment information:</h3>
                    {{ wtf.quick_form(form) }}

            </div>

{% endblock %}

<!-- app/templates/index.html -->
{% extends "base.html" %}

{% block head %}
{{ super() }}
 <link rel="stylesheet" type="text/css" href="https://cdn.jsdelivr.net/npm/cookieconsent@3/build/cookieconsent.min.css" />
{% endblock %}

{% block scripts %}
{{ super() }}
<script src="https://cdn.jsdelivr.net/npm/cookieconsent@3/build/cookieconsent.min.js" data-cfasync="false"></script>
<script>
    window.cookieconsent.initialise({
     "palette": {
         "popup": {
             "background": "#000"
         },
         "button": {
            "background": "#f1d600"
         }
     },
        "theme": "classic"
    });
</script>
{% endblock %}

{% block page_content %}
    <div class="container"><br>
        <div class="col-md-8 col-md-offset-1 col-xs-10 col-xs-offset-1">
        <div id="myCarousel" class="carousel slide center-block"  style="width:900px;">
            <!-- 轮播（Carousel）指标 -->
            <ol class="carousel-indicators">
```

```
                    <li data-target="#myCarousel" data-slide-to="0" class="active"></li>
                    <li data-target="#myCarousel" data-slide-to="1"></li>
                    <li data-target="#myCarousel" data-slide-to="2"></li>
                </ol>

                <!-- （Carousel） Project -->
                <div class="carousel-inner">
                    <div class="item active">
                        <img src="../static/index1.jpg" alt="First slide">
                        <div class="carousel-caption">
                            <!--<h3>Movie Information</h3>
                            <p>bravo!</p>-->
                        </div>
                    </div>
                    <div class="item">
                        <img src="../static/index2.jpg" alt="Second slide">
                    </div>
                    <div class="item">
                        <img src="../static/index3.jpg" alt="Third slide">
                    </div>
                    <div class="item">
                        <img src="../static/index4.jpg" alt="Third slide">
                    </div>
                </div>

                <a href="#myCarousel" class="left carousel-control" data-slide="prev">
                    <span class="glyphicon glyphicon-chevron-left"></span>
                </a>
                <a href="#myCarousel" class="right carousel-control" data-slide="next">
                    <span class="glyphicon glyphicon-chevron-right"></span>
                </a>
            </div>
        </div>
        </div>

        <script>
            $('#myCarousel').carousel({
                interval:2000,
                wrap:true
            }).on('slide.bs.carousel', function () {
              //alert('slide')
            }).on('slid.bs.carousel', function () {
              //alert('slid')
            })
            $('#btn').click(function(){
                //$('#myCarousel').carousel(2);
            });
        </script>

        <div class="center-block">
        <div class=" col-md-7 col-md-offset-2 col-xs-10 col-xs-offset-1">
            <h1>Hello, {% if current_user.is_authenticated %}{{ current_user.username }}{% else %}Stranger{% endif %}!
                Welcome to Gym.</h1>

            <h3>Swimming Pool</h3>
    <p>Our main swimming pool is great for fun and fitness alike. With both shallow and deep ends, this pool is widely accessible for all ages and capabilities. We have a wide variety of swimming lessons for both adults and children as well as a timetable of varied activities.
    </p>
    <h3>Premier Suite (Sauna / Steam room)</h3>
    <p>Our premier suite is the ideal place to unwind and relax after a long day or intense workout. This is an adult only zone, with its own changing area located on the pool level.
    </p>
    <h3>Gym</h3>
    <p>Our gym was fully refurbished and re-opened in October 2012. Because of this the gym now offers a bigger and better fitness experience. The upgrade has seen the gym double in size and has made room for brand new state of the art gym equipment. New flooring, air conditioning and paintwork have been added to compliment the upgrade. This gym offers 40 stations including but not limited to:

    Treadmills
    X trainers
    Bikes
    Rowing machines
    Steppers
    Cable machines

    Selected Technogym equipment have audio jacks that allow you to connect to the in gym entertainment system. Please note: an induction is required for your own safety before using the gym for the first time.
    </p>
```

```
<h3>Sports hall</h3>
<p>Our sports hall has a number of courts available to hire for a variety of sports including badminton, basketball, 5-a-side football and tennis. The venue is also available to hire for non-sporting functions.
</p>
<h3>Outdoor Pitch</h3>
<p>Located outside the centre we a state of the art Floodlit 3rd Generation pitch that is suitable for 5/6 a side matches. This pitch is available for hire - contact the site for prices and availability.
</p>
<h3>Childrens Parties</h3>
<p>Why not celebrate your child's next birthday in style? Our parties are designed to ensure your child's day is packed with fun and yours is stress and hassle free. View our range of parties.
</p>
<h3>Parking</h3>
<p>We have 2 car parks and 6 disabled spots accommodating over 100 vehicles. We also have a undercover cycle rack.</p>
    </div>
    </div>

{% endblock %}


<!-- app/templates/my_booking.html -->
{% extends "base.html" %}

{% block title %}My booking{% endblock %}

{% block page_content %}
    {% if number == 0  %}
        <div class="col-md-offset-4">
            <a href="{{ url_for('main.index') }}">You have no bookings.</a>
        </div>
    {% else %}
        <h4 id="cart">You have {{ number }} booking histories!</h4>
    {% endif %}

<div class="container-fluid">
    {% for booking in bookings %}
        <div class="col-md-10 col-md-offset-1 col-xs-10 col-md-offset-1">
```

```
             <a href="../static/{{booking.id}}.pdf" download="receipt.pdf">Download the receipt</a>
        </div>
            <div class="col-md-10 col-md-offset-1 col-xs-10 col-md-offset-1">
                <table class="table table-striped table-hover mytable" id="complete-table" width='100%' border='0' cellspacing='0' cellpadding='0'  style='table-layout: fixed'>
                    <tbody>
                        <tr>
                            <th colspan='3' style='word-wrap: break-word'>Booking id:{{booking.id}}</th>
                            <th colspan='3' style='word-wrap: break-word'>Price:${{booking.fees}}.0</th>
                            <th colspan='3' style='word-wrap: break-word'>Status:{{booking.status }}</th>
                            <th colspan='3' style='word-wrap: break-word'>Payment:{{booking.payment}}</th>
                            <th colspan='3' style='word-wrap: break-word'>{{booking.timestamp }}</th>
                            {% if booking.status == "Unpaid" %}
                                <th colspan='3' style='word-wrap: break-word'>
                                    <a class="btn btn-mini btn-success" href="{{url_for('.handle_card_booking',book_id=booking.id)}}">Pay now</a>
                                </th>
                                <th colspan='3' style='word-wrap: break-word'>
                                    <a class="btn btn-mini btn-info" href="{{url_for('.cancel_booking',id=booking.id)}}">Cancel membership</a>
                                </th>
                            {% elif  booking.status == "Paid" %}
                                <th colspan='3' style='word-wrap: break-word'>
                                    <a class="btn btn-mini btn-info" href="{{url_for('.cancel_booking',id=booking.id)}}">Cancel membership</a>
                                </th>
                            {% endif %}
                        </tr>
                    </tbody>
                </table>
            </div>
                    {% for timetable in timetable_booking %}
                        {% if booking.time_id == timetable.id %}
                            <div class="row">
```

```
                                <div class="col-md-4 col-md-offset-2 col-xs-6 col-xs-offset-4">
                                <h3>Booking content:</h3>
                                    {% for facility in facilities %}
                                        {% if facility.id == timetable.facility_id %}
                                             <span>Facility:{{ facility.name }} <br> Date: {{timetable.date }}<br>Time:{{timetable.start_time }}:00  ~ {{timetable.end_time }}:00<br></span>
                                        {% endif %}
                                    {% endfor %}

                                </div>
                            </div>
                        {% endif %}
                    {% endfor %}

    {% endfor %}
</div>

{% endblock %}

# migrations/env.py
from __future__ import with_statement
from alembic import context
from sqlalchemy import engine_from_config, pool
from logging.config import fileConfig

config = context.config

fileConfig(config.config_file_name)

from flask import current_app
config.set_main_option('sqlalchemy.url', current_app.config.get('SQLALCHEMY_DATABASE_URI'))
target_metadata = current_app.extensions['migrate'].db.metadata

def run_migrations_offline():
    url = config.get_main_option("sqlalchemy.url")
    context.configure(url=url)

    with context.begin_transaction():
        context.run_migrations()

def run_migrations_online():
    engine = engine_from_config(
                config.get_section(config.config_ini_section),
                prefix='sqlalchemy.',
                        poolclass=pool.NullPool)

    connection = engine.connect()
    context.configure(
                        connection=connection,
                        target_metadata=target_metadata
                        )

    try:
        with context.begin_transaction():
            context.run_migrations()
    finally:
        connection.close()

if context.is_offline_mode():
    run_migrations_offline()
else:
    run_migrations_online()

# migrations/alembic.ini

[alembic]
# template used to generate migration files
# file_template = %%(rev)s_%%(slug)s

# set to 'true' to run the environment during
# the 'revision' command, regardless of autogenerate
# revision_environment = false


# Logging configuration
[loggers]
keys = root,sqlalchemy,alembic

[handlers]
keys = console

[formatters]
keys = generic

[logger_root]
level = WARN
handlers = console
qualname =

[logger_sqlalchemy]
level = WARN
handlers =
qualname = sqlalchemy.engine

```

```
[logger_alembic]
level = INFO
handlers =
qualname = alembic

[handler_console]
class = StreamHandler
args = (sys.stderr,)
level = NOTSET
formatter = generic

[formatter_generic]
format = %(levelname)-5.5s [%(name)s] %(message)s
datefmt = %H:%M:%S

# migrations/script.py.mako

"""${message}

Revision ID: ${up_revision}
Revises: ${down_revision}
Create Date: ${create_date}

"""

# revision identifiers, used by Alembic.
revision = ${repr(up_revision)}
down_revision = ${repr(down_revision)}

from alembic import op
import sqlalchemy as sa
${imports if imports else ""}

def upgrade():
    ${upgrades if upgrades else "pass"}


def downgrade():
    ${downgrades if downgrades else "pass"}

# test/test_basics.py
import unittest
from flask import current_app
from app import create_app, db

class BasicsTestCase(unittest.TestCase):
    def setUp(self):
        self.app = create_app('testing')
        self.app_context = self.app.app_context()
        self.app_context.push()
        db.create_all()

    def tearDown(self):
        db.session.remove()
        db.drop_all()
        self.app_context.pop()

    def test_app_exists(self):
        self.assertFalse(current_app is None)

    def test_app_is_testing(self):
        self.assertTrue(current_app.config['TESTING'])

# test/test_user_model.py
import unittest
import time
from datetime import datetime
from app import create_app, db
from app.models import User, AnonymousUser, Role, Permission


class UserModelTestCase(unittest.TestCase):
    def setUp(self):
        self.app = create_app('testing')
        self.app_context = self.app.app_context()
        self.app_context.push()
        db.create_all()
        Role.insert_roles()

    def tearDown(self):
        db.session.remove()
        db.drop_all()
        self.app_context.pop()

    def test_password_setter(self):
        u = User(password='cat')
        self.assertTrue(u.password_hash is not None)

    def test_no_password_getter(self):
        u = User(password='cat')
        with self.assertRaises(AttributeError):
            u.password

    def test_password_verification(self):
        u = User(password='cat')
        self.assertTrue(u.verify_password('cat'))
        self.assertFalse(u.verify_password('dog'))

    def test_password_salts_are_random(self):
        u = User(password='cat')
        u2 = User(password='cat')
        self.assertTrue(u.password_hash != u2.password_hash)

    def test_valid_confirmation_token(self):
        u = User(password='cat')
        db.session.add(u)
        db.session.commit()
        token = u.generate_confirmation_token()
        self.assertTrue(u.confirm(token))

    def test_invalid_confirmation_token(self):
        u1 = User(password='cat')
        u2 = User(password='dog')
        db.session.add(u1)
        db.session.add(u2)
        db.session.commit()
        token = u1.generate_confirmation_token()
        self.assertFalse(u2.confirm(token))

    def test_expired_confirmation_token(self):
        u = User(password='cat')
        db.session.add(u)
        db.session.commit()
        token = u.generate_confirmation_token(1)
        time.sleep(2)
        self.assertFalse(u.confirm(token))

    def test_valid_reset_token(self):
        u = User(password='cat')
        db.session.add(u)
        db.session.commit()
        token = u.generate_reset_token()
        self.assertTrue(User.reset_password(token, 'dog'))
        self.assertTrue(u.verify_password('dog'))

    def test_invalid_reset_token(self):
        u = User(password='cat')
        db.session.add(u)
        db.session.commit()
        token = u.generate_reset_token()
        self.assertFalse(User.reset_password(token + 'a', 'horse'))
        self.assertTrue(u.verify_password('cat'))

    def test_valid_email_change_token(self):
        u = User(email='john@example.com', password='cat')
        db.session.add(u)
        db.session.commit()
        token = u.generate_email_change_token('susan@example.org')
        self.assertTrue(u.change_email(token))
```

目录

一、 引言

随着时代的发展，民众对于体育锻炼的需求不断地增长。同时信息技术的发展也带来了管理系统的不断进步。本软件使用 Python Flask 框架实现了一个体育中心管理系统，Sqlite 用作体育中心的数据库管理系统。对于体育中心管理系统数据库，确保除管理员外，任何人都无权访问该数据库。管理员可以在该系统中创建活动，管理相关体育设施的预定，以及会员套餐的出售等。普通用户可以购买会员，预定活动等。

二、 软件概述

1. 软件结构

```
.
|   config.py
|   flasky.py
|   requirements.txt
|
├──app
|   |   decorators.py
|   |   email.py
|   |   models.py
|   |   __init__.py
|   |
|   ├──auth
|   |       forms.py
|   |       views.py
|   |       __init__.py
|   |
|   ├──main
|   |       errors.py
|   |       forms.py
|   |       views.py
|   |       __init__.py
|   |
|   ├──static
|   |       favicon.ico
|   |       fitness.jpg
|   |       hall.jpg
|   |       index1.jpg
|   |       index2.jpg
|   |       index3.jpg
|   |       index4.jpg
|   |       pool.jpg
|   |       squash.jpg
|   |       styles.css
|   |
│   └─templates
│       │   403.html
│       │   404.html
│       │   500.html
│       │   base.html
│       │   booking_receipt.html
│       │   booking_receipt.txt
│       │   book_facility.html
│       │   business.html
│       │   cancel_booking.html
│       │   cancel_membership.html
│       │   configure_activity.html
│       │   configure_facility.html
│       │   configure_timetable.html
│       │   edit_profile.html
│       │   facility.html
│       │   facility_info.html
│       │   handle_card_booking.html
│       │   handle_card_membership.html
│       │   index.html
│       │   membership.html
│       │   my_booking.html
│       │   my_card.html
│       │   my_membership.html
│       │   pay_membership.html
│       │   pricing_list.html
│       │   search_booking.html
│       │   search_membership.html
│       │   timetable_all.html
│       │   timetable_facility.html
│       │   user.html
│       │
│       ├─auth
│       │   │   change_email.html
│       │   │   change_password.html
│       │   │   login.html
│       │   │   register.html
│       │   │   reset_password.html
│       │   │   unconfirmed.html
│       │   │
│       │   └─email
│       │           change_email.html
│       │           confirm.html
│       │           reset_password.html
```

```
|          |
|          └──mail
|                  new_user.html
|
├──migrations
|    |    alembic.ini
|    |    env.py
|    |    script.py.mako
|    |
|    └──versions
|              190163627111_account_confirmation.py
|              38c4e85512a9_initial_migration.py
|              456a945560f6_login_support.py
|              56ed7d33de8d_user_roles.py
|              d66f086b258_user_information.py
|
└──tests
          test_basics.py
          test_user_model.py
          __init__.py
```

2. 功能

本程序具有以下功能：

1. 该系统可以查看指定时间/天数范围内所有设施的时间表;
2. 查看指定设施在一定时间/天数范围内的时间表;
3. 支付每月或每年的会员费;
4. 在指定的日期和时间预订并支付活动费用;
5. 取消会员资格;
6. 取消预订;
7. 展示设施图片;
8. 显示设施的定价列表;
9. 银行卡预定付款（模拟）;
10. 存储收据，并按需显示;
11. 生成 PDF 收据;
12. 可以显示收据二维码，用于验证;
13. 支持创建用户帐户和用户登录;
14. 存储用户的支付卡详细信息，以便快速结账;
15. 用户帐户和数据具有良好的安全性;
16. 可以配置设施和活动;
17. 支持邮件确认登录;
18. 此管理系统软件提供了响应灵敏、友好的用户界面。

三、 运行环境

1. 硬件环境

本软件的编写与测试运行在作者本人的个人笔记本电脑上。由于条件限制，作者未测试在其它硬件环境下的运行效果，理论上符合运行 Java 8 虚拟机硬件条件的，有 1G 以上的 RAM 和硬盘存储空间的所有 PC 机上都能运行本程序。

作者本人个人笔记本电脑配置：

CPU: Intel Core i7-8550U 1.80GHZ

RAM: 8GB

硬盘: SSD 128GB 机械 1TB

2. 软件环境

因为本程序采用 Python 3 Flask 框架，因而凡是支持 Python 3 虚拟机的操作系统都可以编译运行本程序。作者的编译运行环境如下：

Ubuntu 18.04（64 位）

Python 3.7.5（64 位）

同时推荐系统安装 virtualenv，便于 Python 包的管理。

Python 库环境如下：

- ***django-qrcode==0.3***
- ***pdfkit==0.6.0***
- ***alembic==0.9.3***
- ***blinker==1.4***
- ***click==6.7***
- ***dominate==2.3.1***
- ***Flask==0.12.2***
- ***Flask-Bootstrap==3.3.7.1***
- ***Flask-Login==0.4.0***
- ***Flask-Mail==0.9.1***
- ***Flask-Migrate==2.0.4***
- ***Flask-Moment==0.5.1***
- ***Flask-SQLAlchemy==2.2***
- ***Flask-WTF==0.14.2***
- ***itsdangerous==0.24***
- ***Jinja2==2.9.6***
- ***Mako==1.0.7***
- ***MarkupSafe==1.0***
- ***python-dateutil==2.6.1***
- ***python-editor==1.0.3***
- ***six==1.10.0***
- ***SQLAlchemy==1.1.11***
- ***visitor==0.1.3***
- ***Werkzeug==0.12.2***
- ***WTForms==2.1***

下面是详细开发环境描述：

类型	描述
网络浏览器	Microsoft edge, Google chrome 支持分辨率: 800 X 600 和 1024 X 768
应用服务器	Flask 和 Django 提供的内置服务器 RESTful API
Python 开发环境	JetBrains PyCharm 2.3
Python 版本	Python3
数据库	SQLite
配置管理工具	VSS (Offshore)
缺陷记录工具	IPM+, Excel sheets
版本控制	Gitlab
组件	Struts 2.0 (including Ajax) EJB 3.0 JPA Log4j

四、 技术细节

1. 解决方案摘要

所有用户都需要用他们的电子邮件地址注册一个帐户。但是，员工和客户会有不同的 web 界面来支持不同的功能。

我们的第一个目标用户是有体育锻炼需求的个人。人们将使用本网站加入会员，查看体育中心支持的所有设施和活动的时间表和信息，查看每个活动的价目表，在线预订每个设施，他们还将收到收据。

我们的第二个用户是体育中心的工作人员，他们将使用这个网站来管理体育中心。他们可以更新最新的信息，如时间表和设施的可用性，他们希望为体育中心的客户预订定期会议。他们还将处理信用卡和现金支付，打印收据和门票。

2. 数据建模

Role Model			
Column Name	Description	Column Data Type	Column Null Option
id	role id	integer	Not null
name	role name	string	Not null
default		boolean	Not null
permissions		integer	Not null
users			

Membership_type Model			
Column Name	Description	Column Data Type	Column Null Option
id	type id	integer	Not null
length	months	integer	Not null
price		integer	Not null
memberships			Not null

Membership Model			
Column Name	Description	Column Data Type	Column Null Option
id	memnberhsip id	integer	Not null
title		string	Not null
firstname		string	Not null
lastname		string	Not null
status		string	Not null
start_date		datetime	Not null
end_date		datetime	Not null
membership_type_id	foreign key of membership_type	integer	Not null
account_id	foreign key of account	integer	Not null

User Model			
Column Name	Description	Column Data Type	Column Null Option
id	user id	integer	Not null
email	user register email	string	Not null
username		string	Not null
role_id	user's role	integer	Not null
password_hash	user's set password	string	Not null
confirmed	status of user 's account	boolean	Not null

Facility Model			
Column Name	Description	Column Data Type	Column Null Option
id	facility id	integer	Not null
name	facility's name	string	Not null
capacity	facility's capacity	integer	null
description	description of a facility	string	Not null
url	image url	string	
times			
activities			

Booking Model			
Column Name	Description	Column Data Type	Column Null Option
id	booking id	integer	Not null
number	number of people	integer	Not null
activity	activity name	string	null
status	paid or unpaid	string	Not null
payment	cash or card	string	Not null
fees	price of booking	integer	Not null
timestamp	time of booking	datetime	Not null
time_id	belongs to which timetable	integer	Not null
account_id	belongs to which account	integer	Not null
activity_id	foreginkey ofactivity	integer	Not null

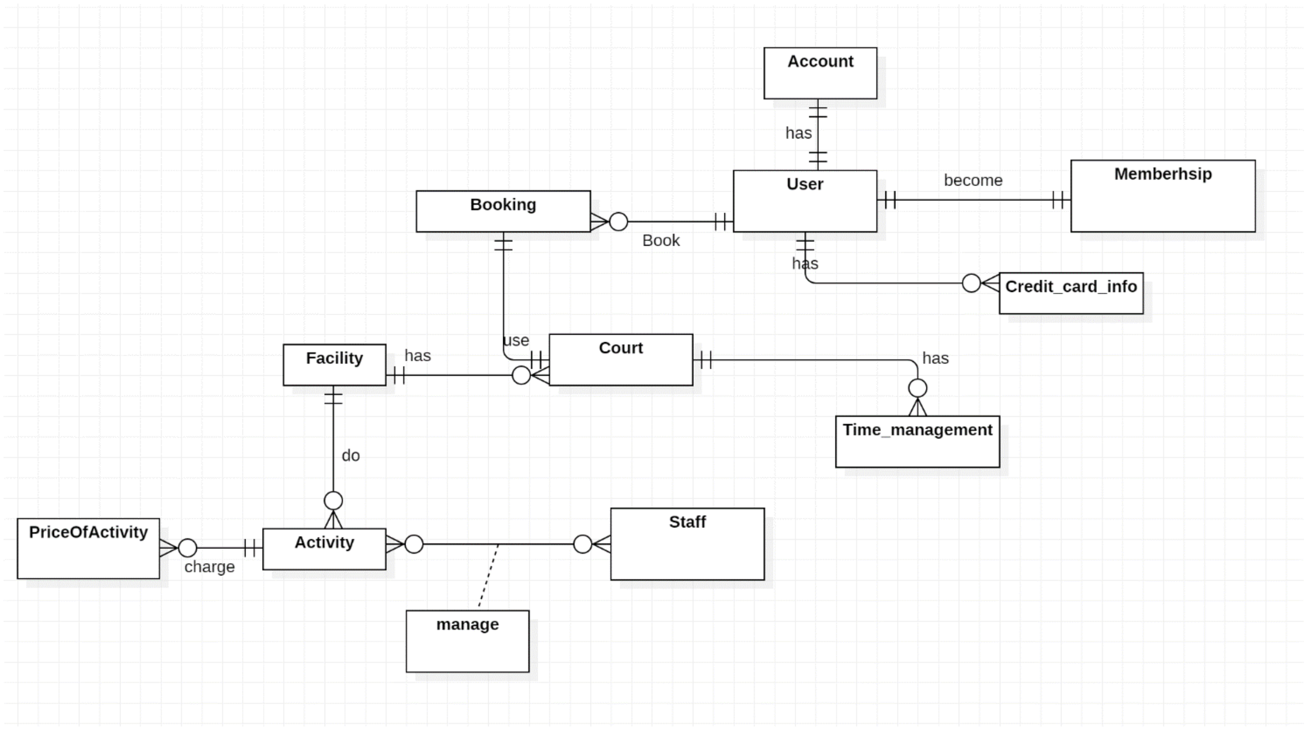
Time_management Model			
Column Name	Description	Column Data Type	Column Null Option
id	booking id	integer	Not null
start_time	session for booking start time	datetime	null
end_time	session for booking end time	datetime	null
date	date	date	null
current_capacity	current capacity	integer	Not null
bookings			
facility_id	belongs to which facility	integer	Not null

Activity Model			
Column Name	Description	Column Data Type	Column Null Option
id	activity id	integer	Not null
activity_name	activity name	string	Not null
repeat_every		integer	null
facility_id	belongs to which facility	integer	Not null
weekly_income	amount of income	integer	
weekly_usage	number of people	integer	
activity_price	price	integer	Not null
bookings			

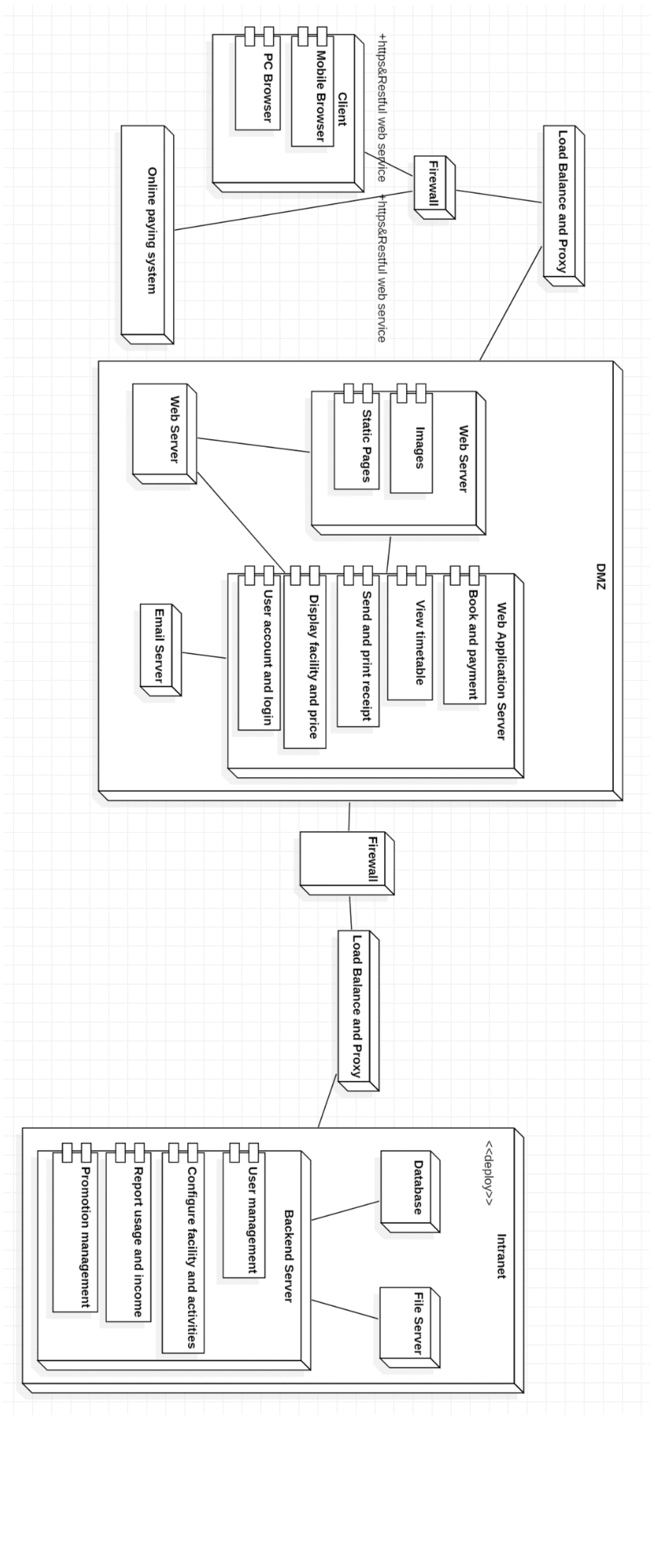
Credit_card_info Model			
Column Name	Description	Column Data Type	Column Null Option
id	credit card id	integer	Not null
card_number	credit card number	integer	Not null
expire_month		integer	Not null
expire_year		integer	Not null
security_code	credit card security code	integer	Not null
holder_name	credit card holder's name	integer	Not null
account_id	belongs to which account	integer	Not null

Account Model			
Column Name	Description	Column Data Type	Column Null Option
id	account id	integer	Not null
user_id	belongs to which user	integer	Not null
bookings			
cards			
membership			

3. 数据库设计



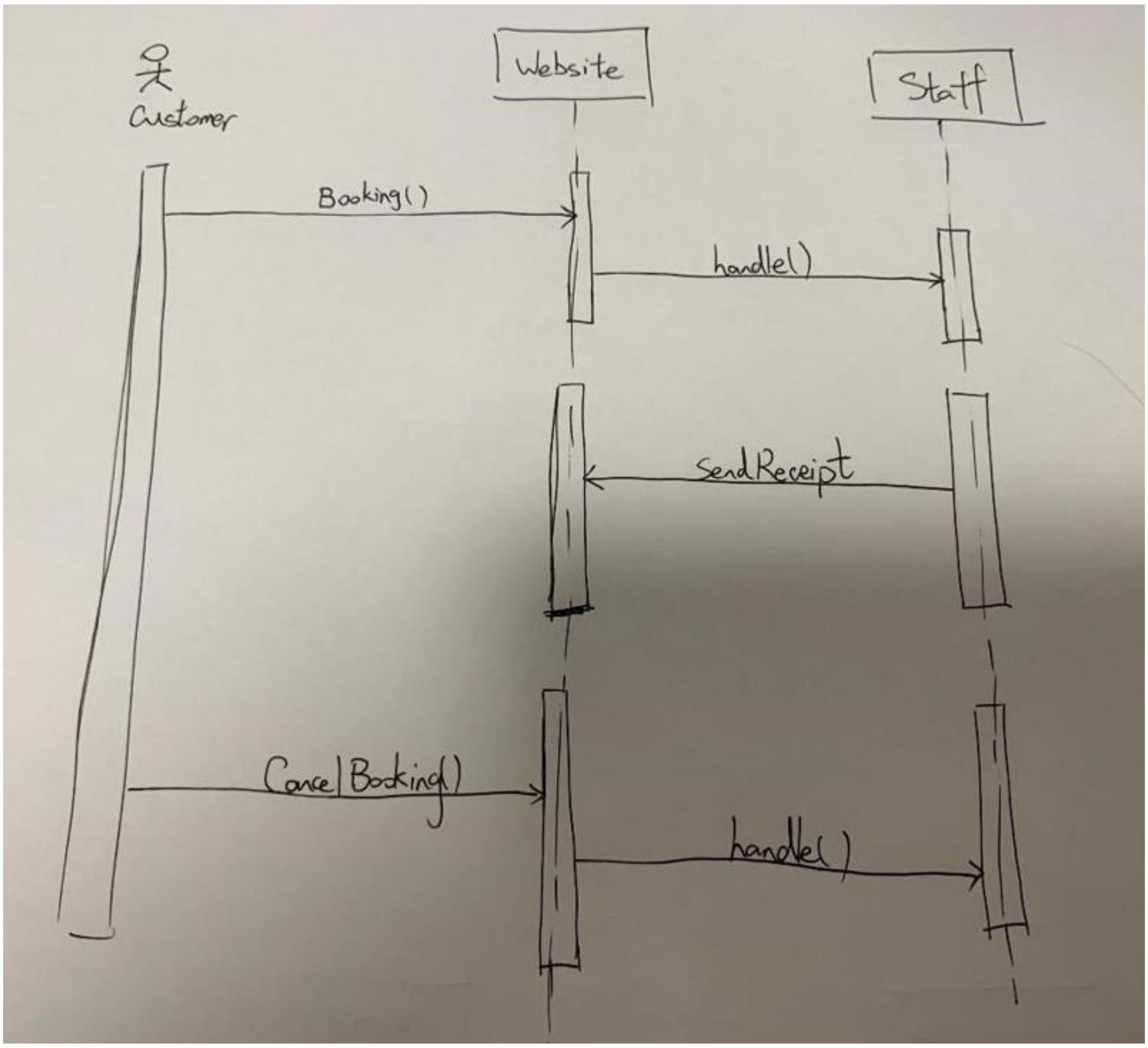
4. 部署方案


Client
Mobile Browser
PC Browser
+https&Restful web service
Firewall
+https&Restful web service
Online paying system
Load Balance and Proxy
DMZ
Web Server
Images
Static Pages
Web Server
Web Application Server
Book and payment
View timetable
Send and print receipt
Display facility and price
User account and login
Email Server
Firewall
Load Balance and Proxy
Intranet
<<deploy>>
Database
File Server
Backend Server
User management
Configure facility and activities
Report usage and income
Promotion management

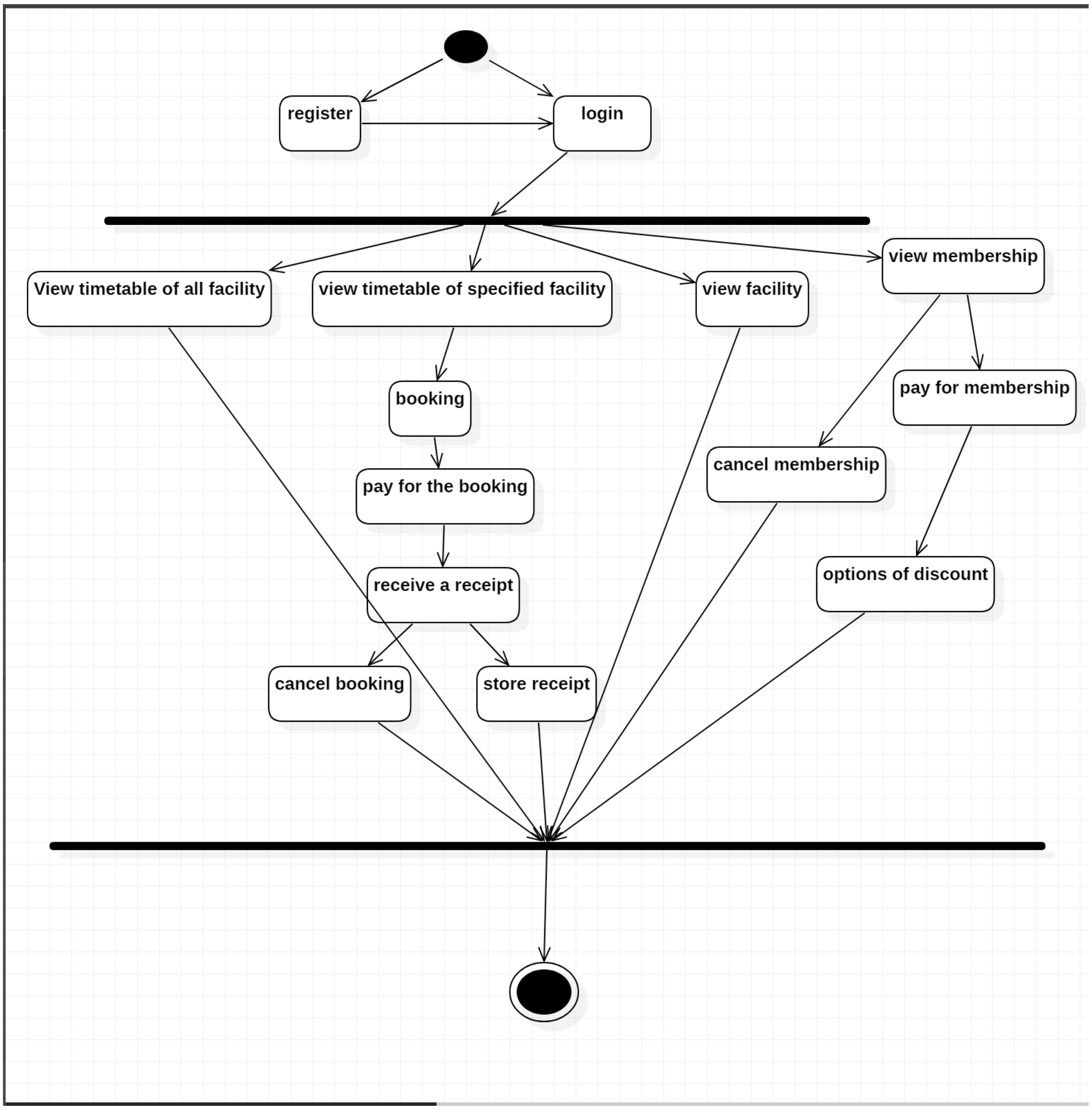
硬件包括智能手机、各种服务器、防火墙和工作负载平衡。该软件包括客户端应用程序/浏览器、DMZ 中的 web 服务器、web 应用服务器、数据库、应用管理后端服务器和电子邮件服务器。

基于 Python Flask 框架实现。采用 Restful Webservice 实现服务器端功能。另外，在这个项目中，移动和网站界面都应该实现，这样 Restful webservice 将是一个不错的选择。负载平衡和服务器集群技术是保证可靠性和防止故障转移的必然选择。为了增强安全性和隐私性，我们设计了两个防火墙。此外，还将采用响应式设计，允许用户使用来自不同设备的应用程序，包括智能手机和平板电脑，这将提高可用性。

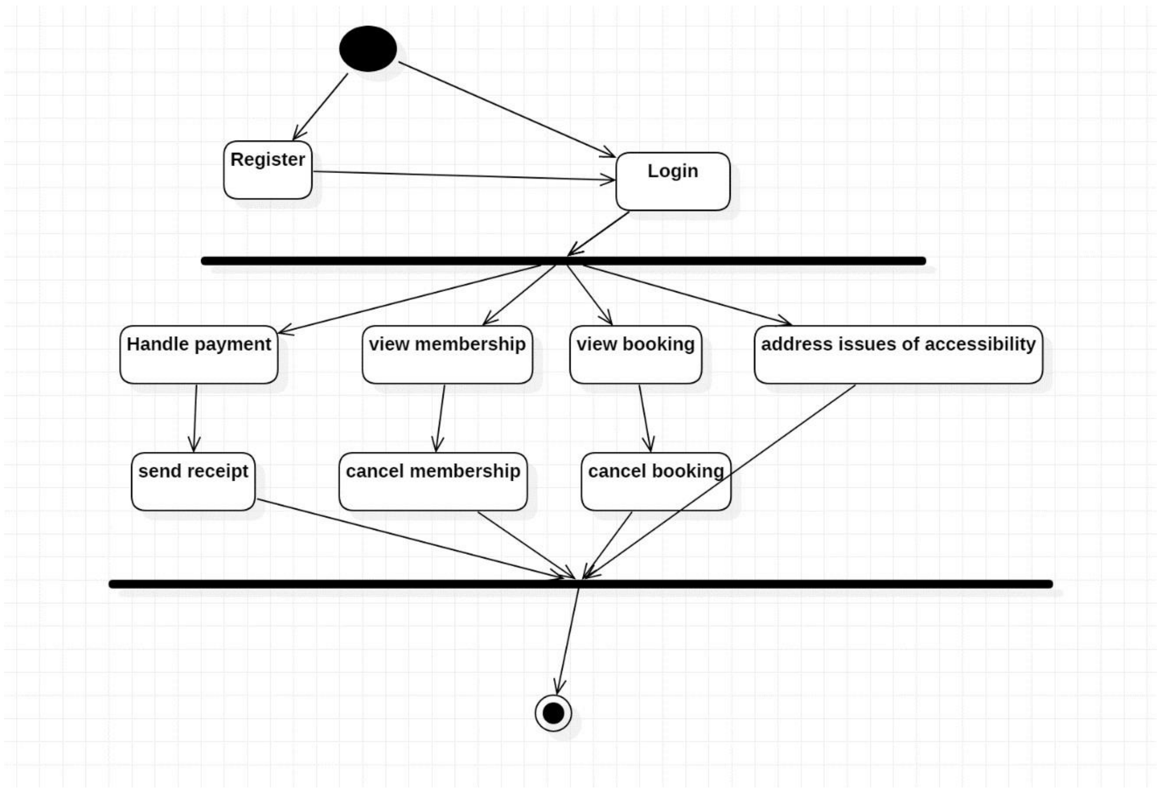
5. 用户使用方案



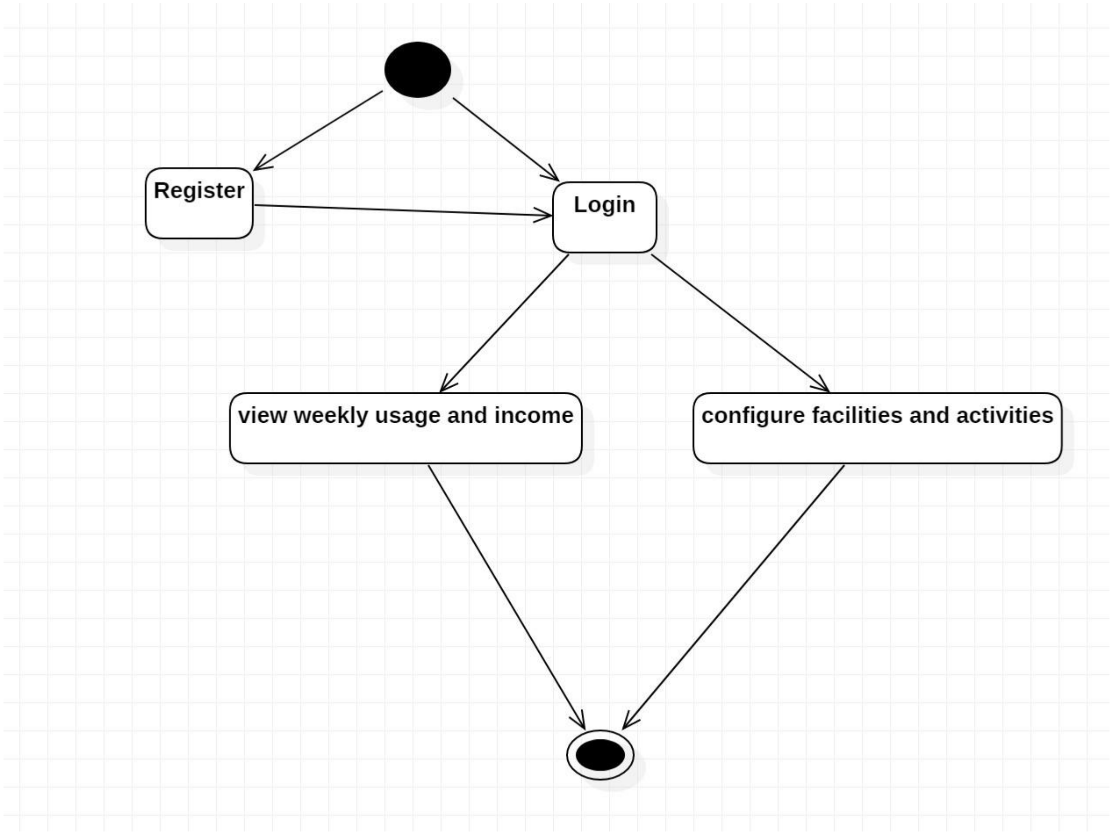
我是小明，我是体育中心的会员。我想打篮球，所以我访问了体育中心的网站。我想看一下篮球场的时间表，以便知道我什么时候可以去打篮球。我想订一个篮球场，以便有一天我能打篮球。



我是小明，是体育中心的顾客。我想看看所有设施的时间表，以便我可以选择什么时间和什么样的运动。我想查看具体设施的时间表，以便选择预定场地的时间。我想取消预订，因为我有事，这样钱就不会浪费了。我想参观一下，这样我就可以下定决心加入会员行列了。我想成为会员，这样我就有一些折扣的权利。



我是小红，是体育中心的工作人员。我想办理付款手续，以便处理客户的预订。我想寄出收据，以便顾客能使用他们的预订。我想查看客户的会员资格，以便我可以告诉客户其会员信息。我要能够查看预定，这样我就可以为顾客取消预订。我想解决访问的问题，这样我们的网站就不会有问题。


Register
Login
view weekly usage and income
configure facilities and activities

我是小李，也是体育中心的经理。我想查看每周的使用情况和收入，以便我能管理体育

中心的业务状况。我想配置设施和活动，以便客户了解我们最新的设施和活动。

6. 实现机制

1. 数据管理

数据库将通过 Flask web 应用程序访问。SQAlchemy 将用于管理数据库连接和数据库操作，如选择、插入、删除和更新。

2. 事务管理

数据库事务管理器用于管理本地数据库事务。事务由业务层控制，而数据库操作则在持久层上执行。

体育中心管理系统中的事务使用数据库事务管理器处理。

3. 会话管理

为每个登录用户维护 HTTP 会话。应用程序中的所有操作都将通过公共方法验证有效会话的存在。

对于可伸缩性，将确保会话数据最小。

会话超时将在应用程序服务器上配置。会话超时后，用户将被转发到登录页。

4. 分页

分页用于显示或维护项目列表的屏幕。这将通过仅检索显示所需的数据来改进用户界面和系统性能。每页显示的记录将是可配置的项目。

5. 排序

默认排序函数使用要排序的列 id 和当前已排序的列 id 来显示排序指示符，还附加必需的 JavaScript。

6. 错误和异常处理

应用程序可能会遇到逻辑或系统错误。验证错误等异常被分类为逻辑错误，并将显示在同一页中。系统错误（如数据库不可用导致的错误）将显示在不同的错误页中。

错误将使用 Log4j 记录到错误日志文件中。错误日志文件的位置和名称将在应用程序属性文件中配置。

7. 业务验证和错误报告

- **客户端验证**

所有数据类型验证都将使用 JavaScript 完成，并使用标准报告。

- **服务器端验证**

当用户执行保存和提交操作时，需要进行业务验证。

验证/错误报告将在整个应用程序中以一致的方式处理。

8. 日志记录和跟踪

所有应用程序错误和跟踪语句都将记录到一个文件中。Log4j 组件将用于此目的。对于所有异常和跟踪，完整的异常和跟踪将写入文件。可以在属性文件中指定每个日志文件的最大大小和最大日志文件数。

9. 可维护性

为了确保应用程序是可维护的，应用程序设计/开发遵循 Java 准则，Sun Java 编码准则。

应用程序构建将使用分层方法（表示、业务和数据）来提高可维护性。

应用程序还将提供可配置的不同级别的日志记录和跟踪。

10. 安全

系统将使用 AES 加密方法对每个用户的密码进行加密。

当用户首次登录时，系统会对用户的密码进行加密，并与该用户存储的加密密码进行比较，如果相同，用户可以登录并将用户对象存储在会话中，否则用户无法登录并记录信息，

在每个 JSP 页面中，首先从会话中获取用户 id，如果在会话中找不到用户 id 或用户无权查看该页，则系统将重定向到登录页。

11. 国际化

在第一个登录页面中，我们可以添加一个下拉列表，用户可以在其中选择中英文之间的语言，应用程序将在会话中存储语言代码。因此，在每个页面中，应用程序将根据会话中存储的语言代码显示属性文件中的文本消息。

12. 网络服务

由于最终用户可能需要从其他网站查询自己的积分和更改自己的密码等，因此集中会员系统提供了网站服务和界面，让其他应用程序调用。

集中式会员系统使用 Metro1.2（一个 Web 服务框架，为最终用户和中间件开发人员提供开发 Web 服务解决方案的工具和基础设施），需要 JDK1.5Update2 或更高版本，由 SUN Microsystem 开发。

13. 设计模式

MVC-应用程序将使用基于 MVC 设计模式的 Struts2。

Decorate-应用程序将使用 Decorate 模式来组织具有许多相同方法的类。

Façade-所有的业务逻辑都将封装在 EJB3 中。因此，系统将提供类作为许多 EJB 方法的 façade 外观。

Factory 方法-系统可以使用 Factory 方法来创建许多对象，而不是直接使用构造函数方法。

3. public void newWord(String choice) 沿袭了 word 类中的 readWord 方法，采用逻辑判断等使其更符合 UI 程序的要求。

4. public void judgeWord() 同样沿袭了 word 类中的 judge 方法，采用逻辑判断等使其更符合 UI 程序的要求。

五、 用户指南

1. 安装软件并启动

在该软件的代码库下，打开终端，确保已经安装了 virtualenv，然后输入以下命令：

```
virtualenv venv && source venv/bin/activate
pip install -r requirements.txt
```

此时 Python 库环境配置成功。

随后，如果你首次使用本软件，还需要进行数据库的初始化操作：

```
export FLASK_APP=flasky.py
$ flask db upgrade
```

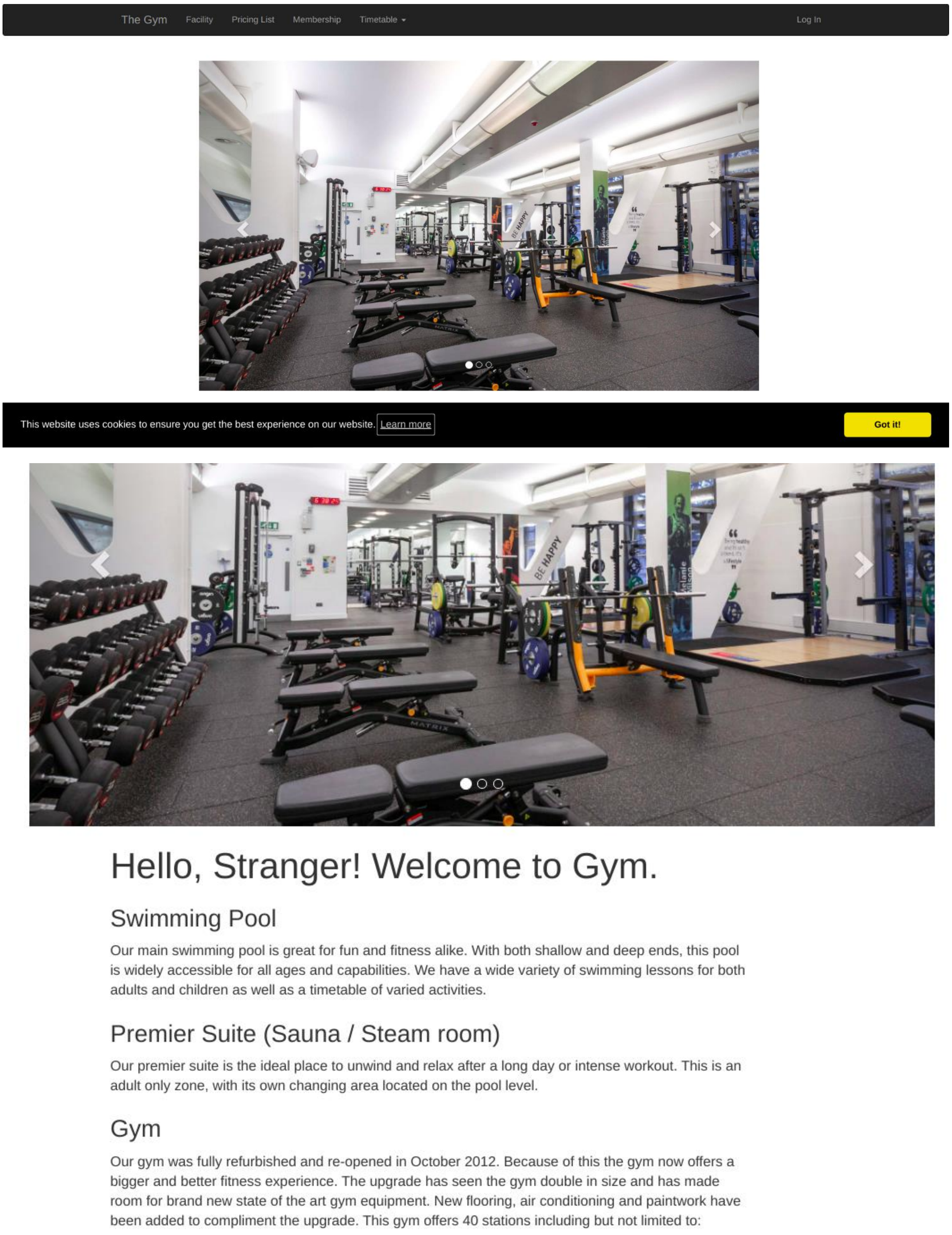
然后打开 flask shell，输入 models.Role.insert_roles()来将角色 ID 添加至数据库中。

最后，选择一个管理员邮箱账号，执行以下命令，服务器便配置完成：

```
export FLASKY_ADMIN=< email 地址>
flask run
```

2. 注册账号

进入首页，点击网站下端的“Got it!”接受 Cookies：



This website uses cookies to ensure you get the best experience on our website.

Hello, Stranger! Welcome to Gym.

Swimming Pool

Our main swimming pool is great for fun and fitness alike. With both shallow and deep ends, this pool is widely accessible for all ages and capabilities. We have a wide variety of swimming lessons for both adults and children as well as a timetable of varied activities.

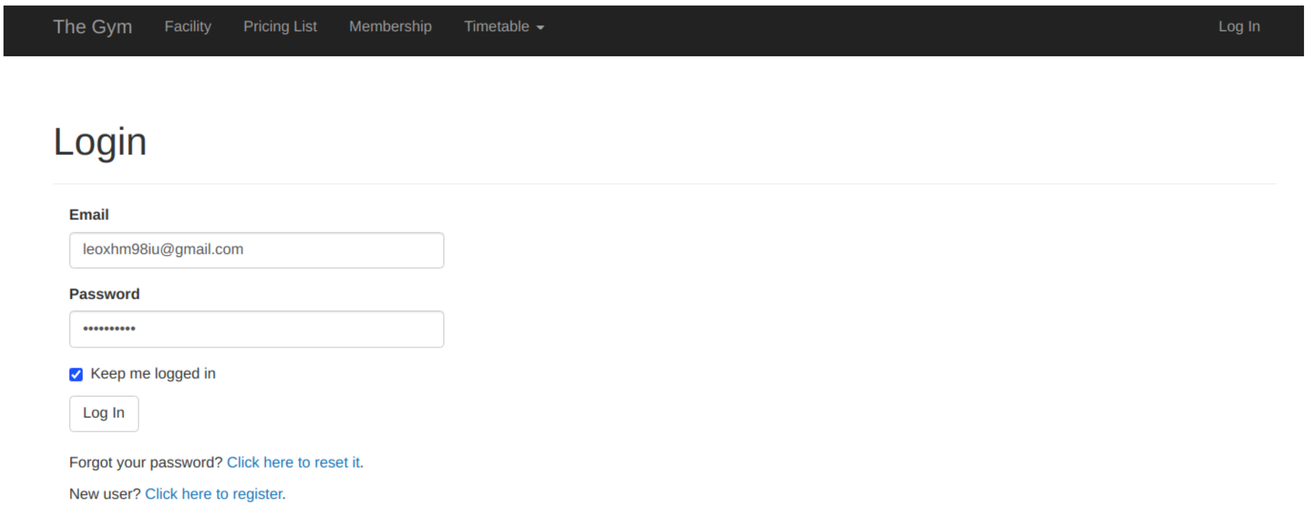
Premier Suite (Sauna / Steam room)

Our premier suite is the ideal place to unwind and relax after a long day or intense workout. This is an adult only zone, with its own changing area located on the pool level.

Gym

Our gym was fully refurbished and re-opened in October 2012. Because of this the gym now offers a bigger and better fitness experience. The upgrade has seen the gym double in size and has made room for brand new state of the art gym equipment. New flooring, air conditioning and paintwork have been added to compliment the upgrade. This gym offers 40 stations including but not limited to:

点击右上角的 Log in，打开用户登录界面：



Login

Email
leoxhm98iu@gmail.com

Password
••••••••••

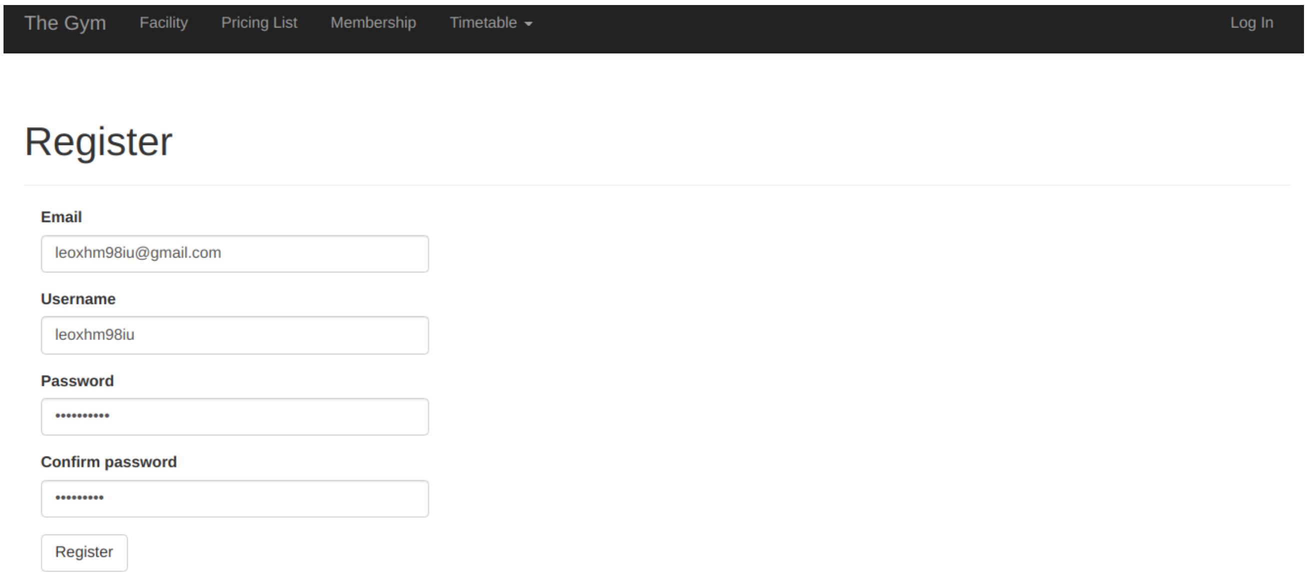
☑ Keep me logged in

Log In

Forgot your password? Click here to reset it.

New user? Click here to register.

点击“Click here to register”注册账号：



Register

Email
leoxhm98iu@gmail.com

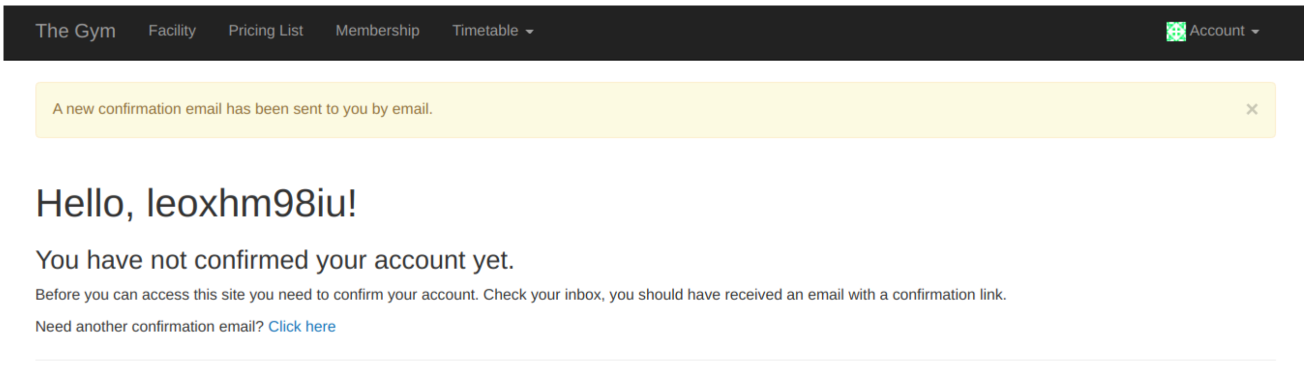
Username
leoxhm98iu

Password
••••••••••

Confirm password
•••••••••

Register

输入注册信息，点击“Register”注册，一封确认邮件将发送到你的账户：



A new confirmation email has been sent to you by email.

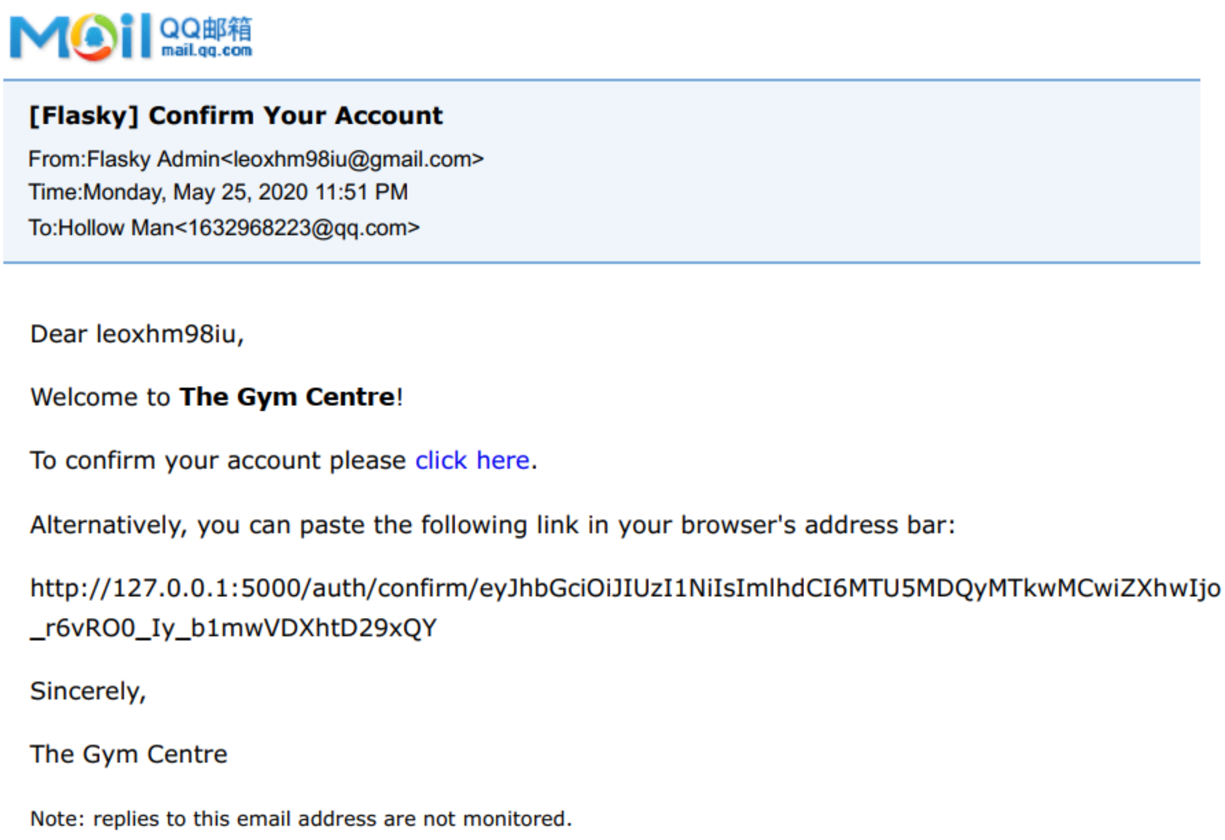
Hello, leoxhm98iu!

You have not confirmed your account yet.

Before you can access this site you need to confirm your account. Check your inbox, you should have received an email with a confirmation link.

Need another confirmation email? Click here

点击邮箱中的链接，确认邮件，注册完成：



[Flasky] Confirm Your Account

From:Flasky Admin<leoxhm98iu@gmail.com>
Time:Monday, May 25, 2020 11:51 PM
To:Hollow Man<1632968223@qq.com>

Dear leoxhm98iu,

Welcome to **The Gym Centre**!

To confirm your account please click here.

Alternatively, you can paste the following link in your browser's address bar:

http://127.0.0.1:5000/auth/confirm/eyJhbGciOiJIUzI1NiIsImlhdCI6MTU5MDQyMTkwMCwiZXhwIjo_r6vRO0_Iy_b1mwVDXhtD29xQY

Sincerely,

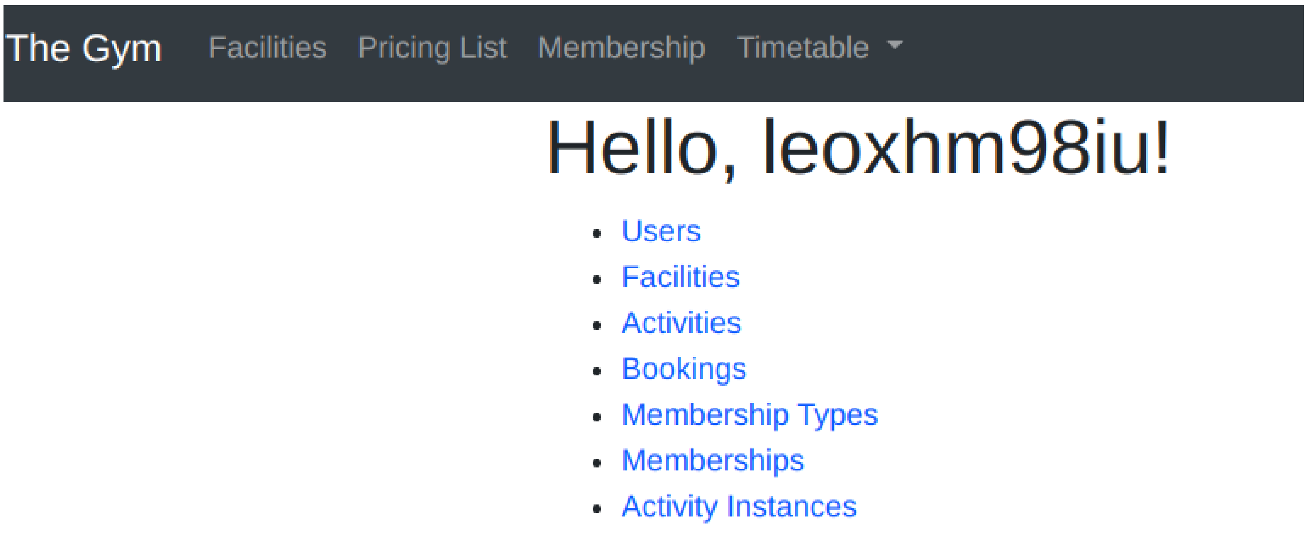
The Gym Centre

Note: replies to this email address are not monitored.

账户类型取决于你的配置，如果你注册用的邮箱为你的管理员邮箱账号，注册成功后会自动拥有管理员权限，否则为普通账户。

3. 管理员

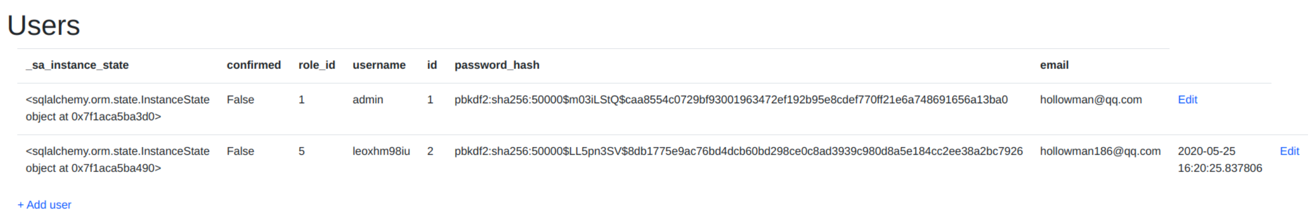
管理员拥有对整个网站的管理权限，下面是管理员可用选项：



Hello, leoxhm98iu!

- Users
- Facilities
- Activities
- Bookings
- Membership Types
- Memberships
- Activity Instances

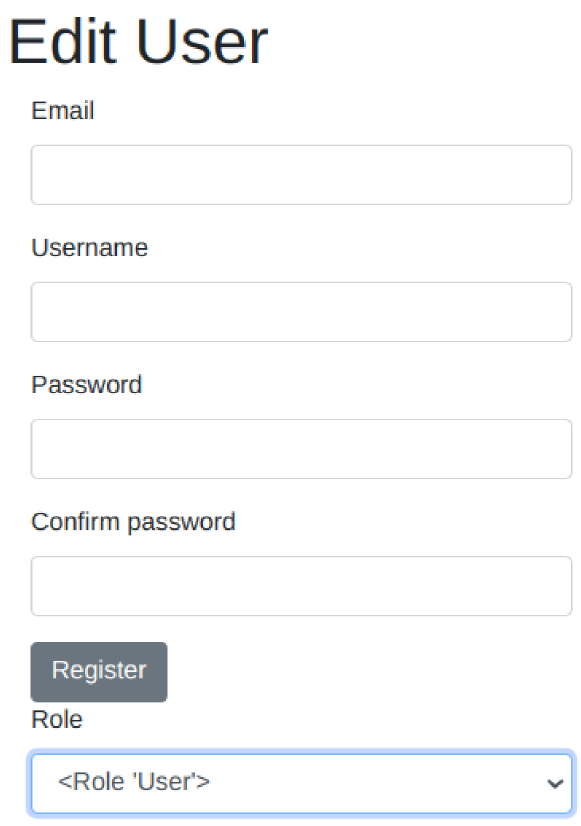
点击 Users，可以查看用户信息，以及手动管理用户：



Users

_sa_instance_state	confirmed	role_id	username	id	password_hash	email		
<sqlalchemy.orm.state.InstanceState object at 0x7f1aca5ba3d0>	False	1	admin	1	pbkdf2:sha256:50000$m03iLStQ$caa8554c0729bf93001963472ef192b95e8cdef770ff21e6a748691656a13ba0	hollowman@qq.com	Edit	
<sqlalchemy.orm.state.InstanceState object at 0x7f1aca5ba490>	False	5	leoxhm98iu	2	pbkdf2:sha256:50000$LL5pn3SV$8db1775e9ac76bd4dcb60bd298ce0c8ad3939c980d8a5e184cc2ee38a2bc7926	hollowman186@qq.com	2020-05-25 16:20:25.837806	Edit

+ Add user



Edit User

Email

Username

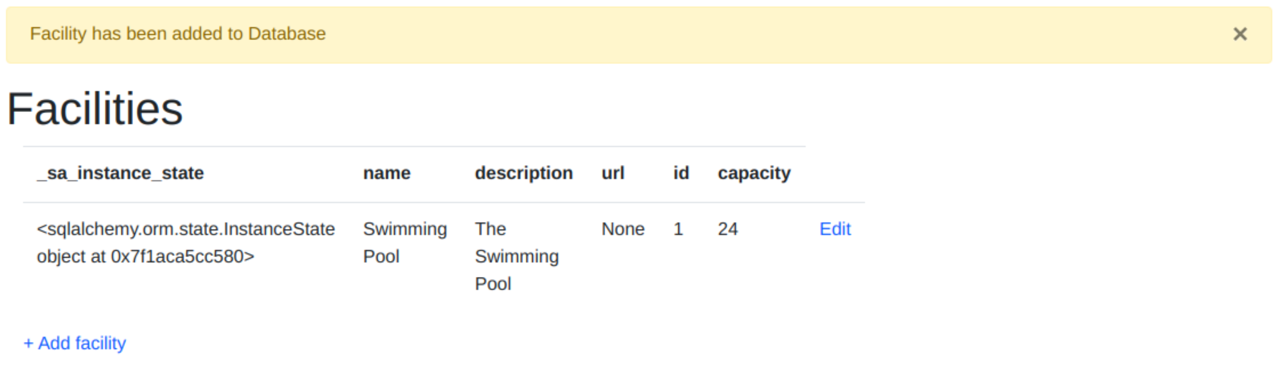
Password

Confirm password

Role

<Role 'User'>

点击 Facilities，可以查看设施信息，以及手动管理设施信息：

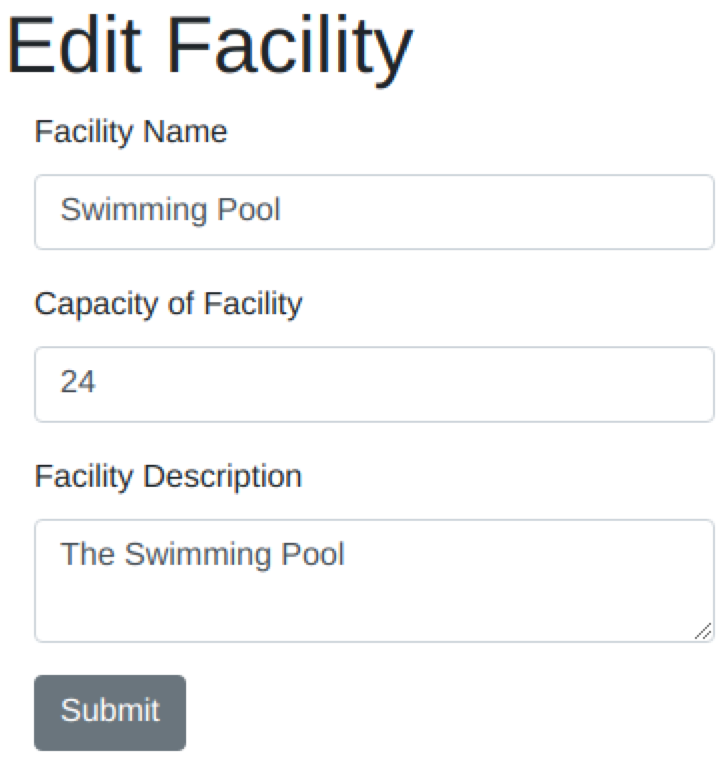


Facility has been added to Database ×

Facilities

_sa_instance_state	name	description	url	id	capacity	
<sqlalchemy.orm.state.InstanceState object at 0x7f1aca5cc580>	Swimming Pool	The Swimming Pool	None	1	24	Edit

+ Add facility



Edit Facility

Facility Name

Swimming Pool

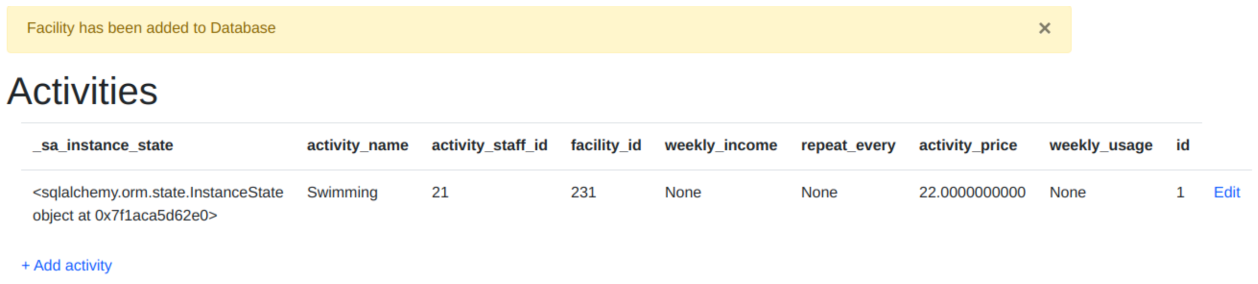
Capacity of Facility

24

Facility Description

The Swimming Pool

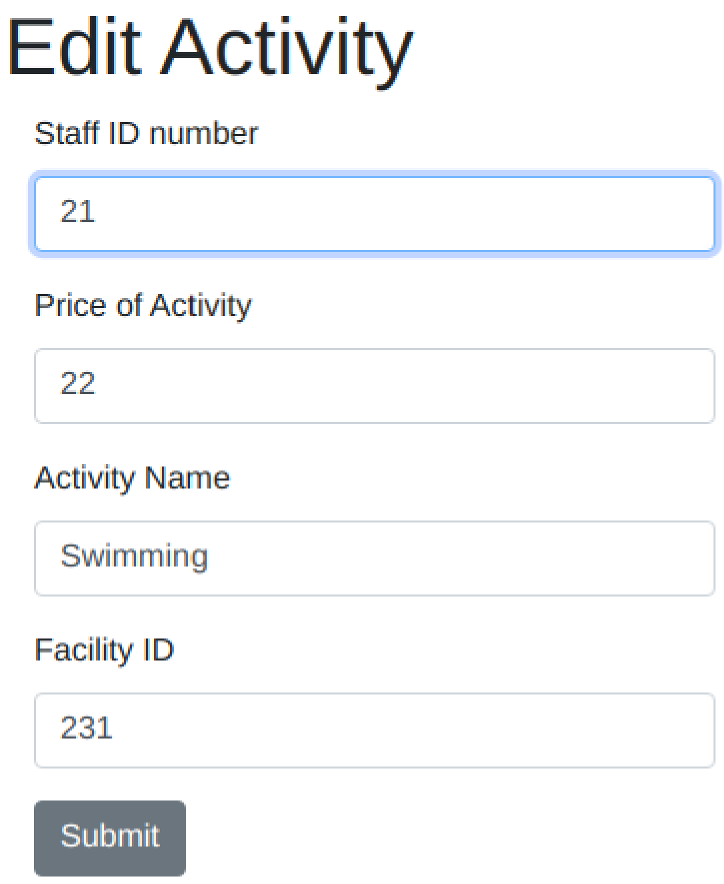
点击 Avtivities，可以查看活动信息，以及手动管理活动：



Facility has been added to Database ×

Activities

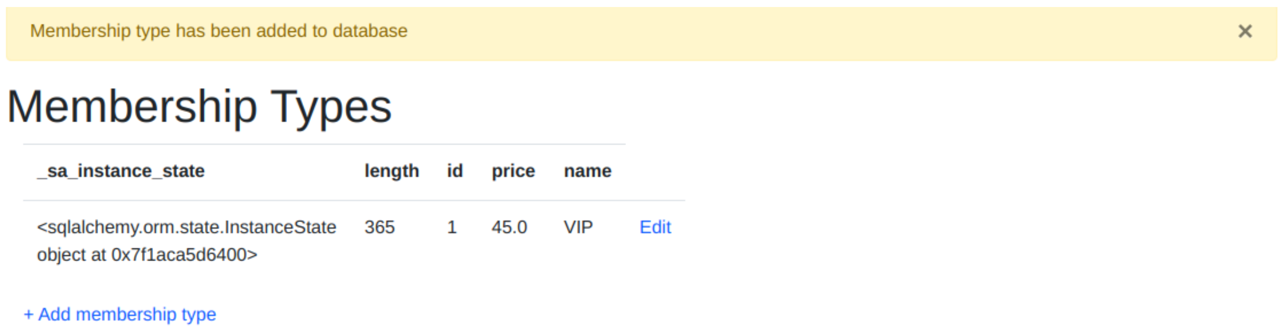
_sa_instance_state	activity_name	activity_staff_id	facility_id	weekly_income	repeat_every	activity_price	weekly_usage	id	
<sqlalchemy.orm.state.InstanceState object at 0x7f1aca5d62e0>	Swimming	21	231	None	None	22.0000000000	None	1	Edit



Edit Activity

Staff ID number	21
Price of Activity	22
Activity Name	Swimming
Facility ID	231

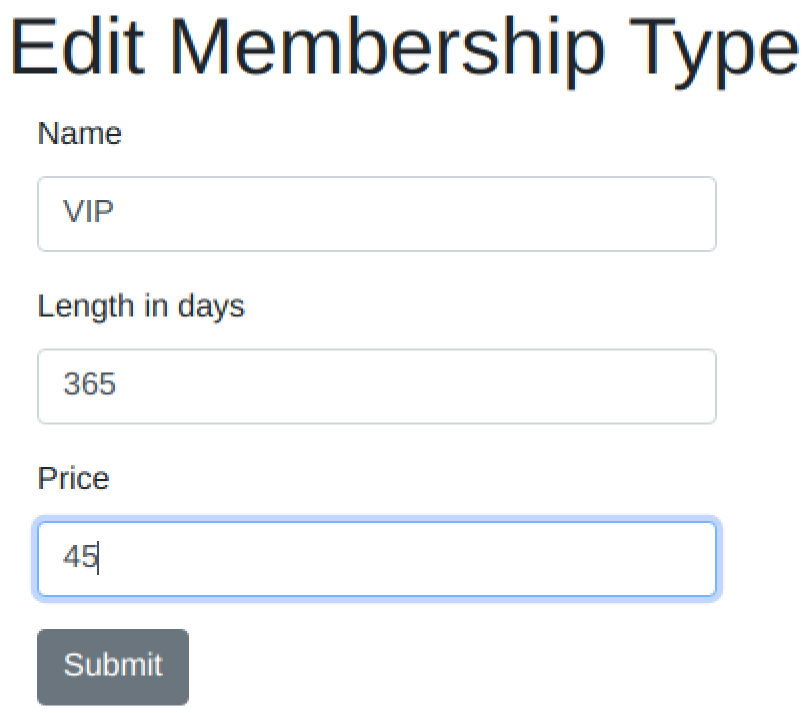
点击 Menbership Types，可以查看会员种类，以及手动管理会员种类：



Membership type has been added to database

Membership Types

_sa_instance_state	length	id	price	name	
<sqlalchemy.orm.state.InstanceState object at 0x7f1aca5d6400>	365	1	45.0	VIP	Edit


Edit Membership Type

Name

VIP

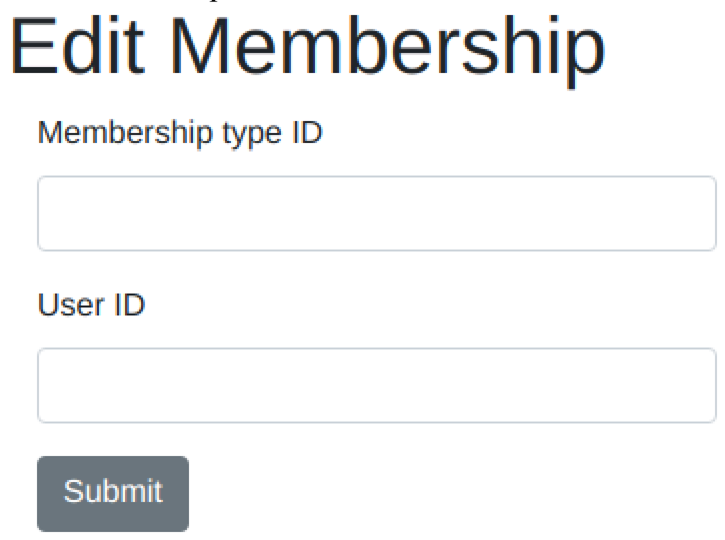
Length in days

365

Price

45

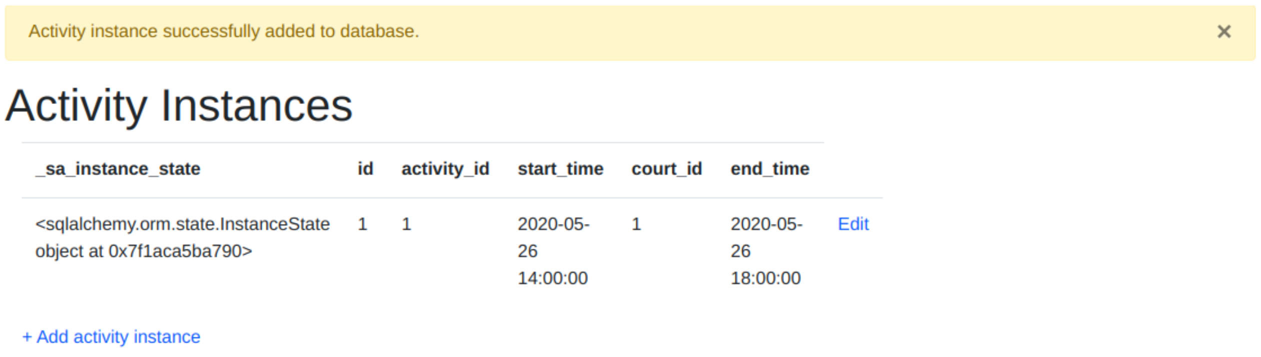
点击 Menbership，可以以及手动管理会员：


Edit Membership

Membership type ID

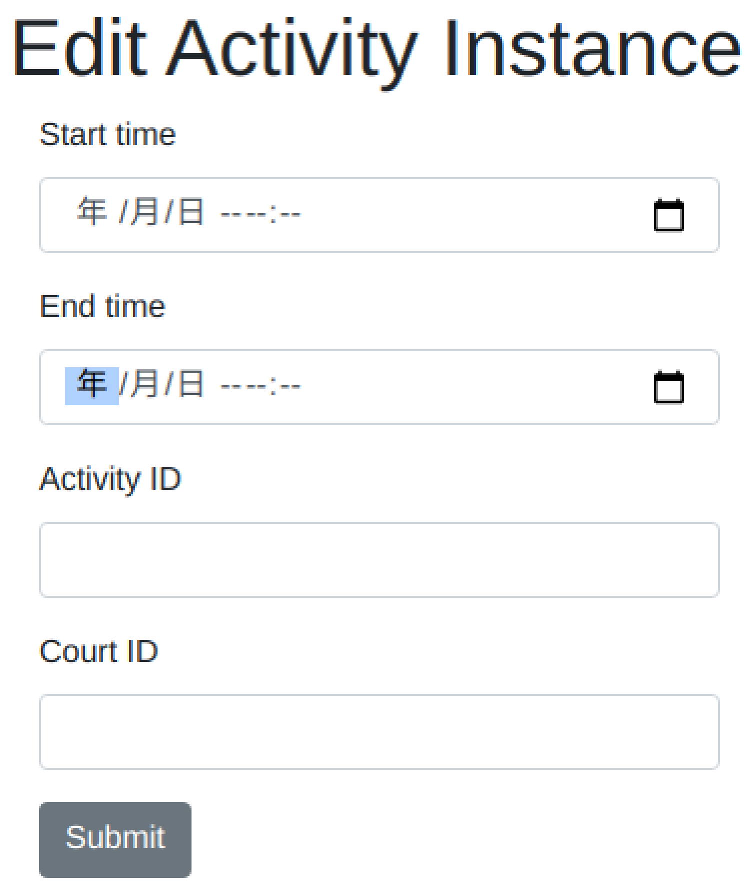
User ID

点击 Activity Instances，可以查看已经发布的活动，以及手动发布活动：


Activity instance successfully added to database.

Activity Instances

_sa_instance_state	id	activity_id	start_time	court_id	end_time	
<sqlalchemy.orm.state.InstanceState object at 0x7f1aca5ba790>	1	1	2020-05-26 14:00:00	1	2020-05-26 18:00:00	Edit


Edit Activity Instance

Start time

年 /月/日 --:--

End time

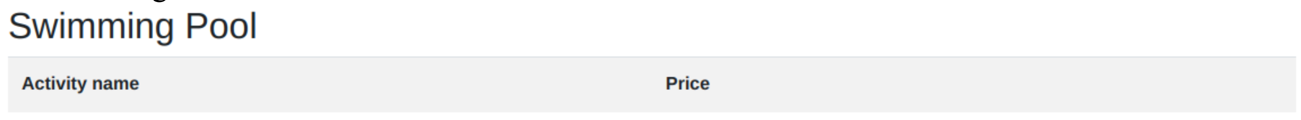
年 /月/日 --:--

Activity ID

Court ID

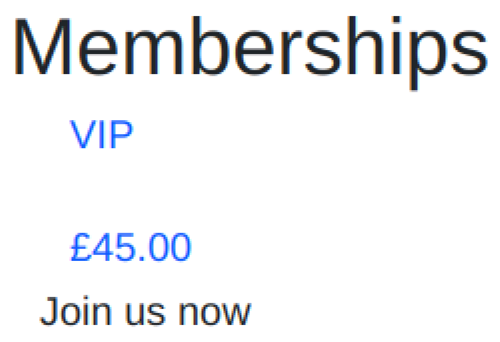
4. 普通用户

用户可以点击上方菜单栏的 Facilities 查看设施列表。
点击 Pricing List 查看活动报价：


Swimming Pool

Activity name	Price

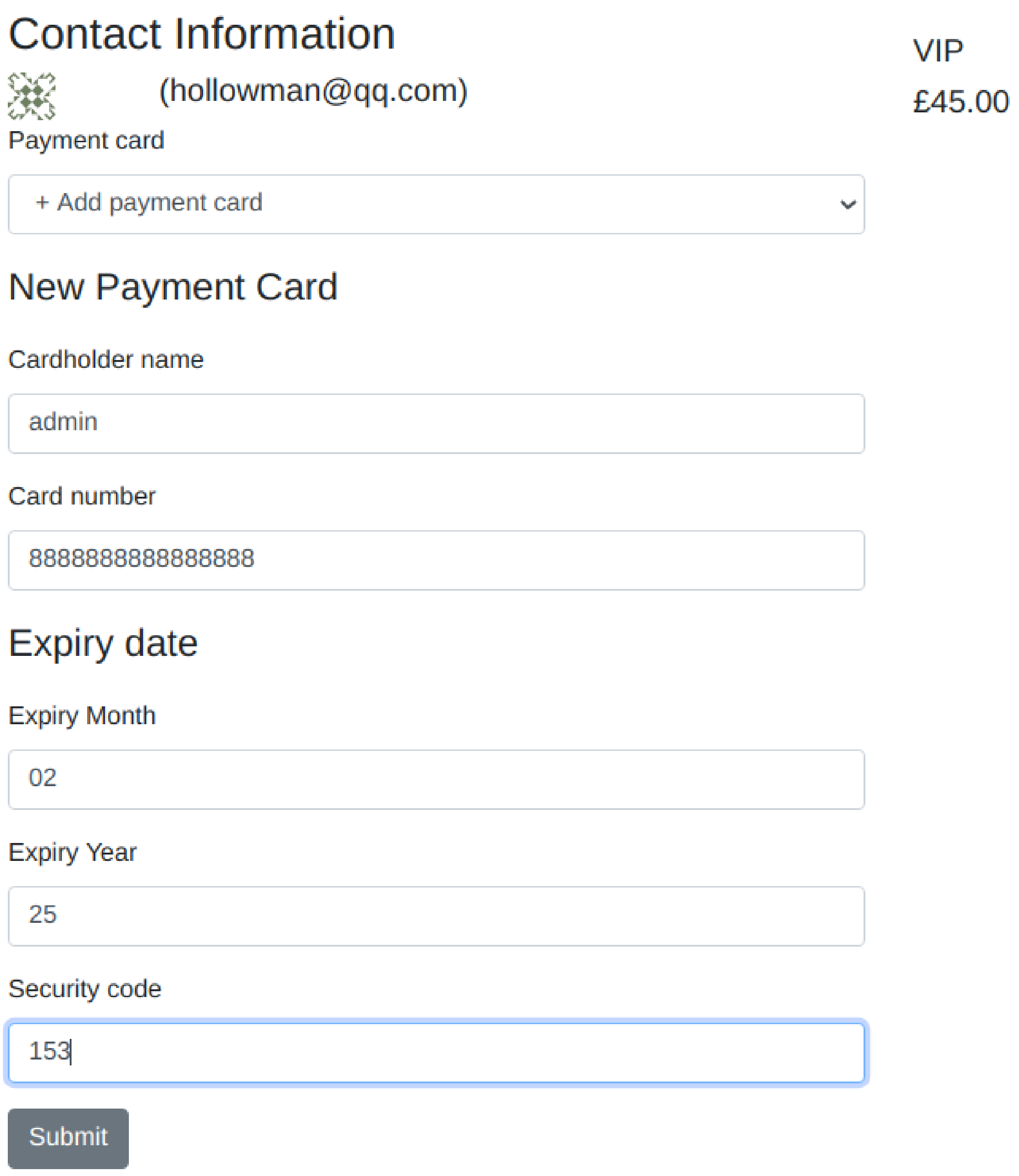
点击 Membership，打开选择会员类型页面，选中你心仪的会员类型：


Memberships

VIP

£45.00

进入支付界面，如果你还没有绑定银行卡，这里可以添加你的银行卡信息：


Contact Information

(hollowman@qq.com)

Payment card

+ Add payment card

New Payment Card

Cardholder name

admin

Card number

8888888888888888

Expiry date

Expiry Month

02

Expiry Year

25

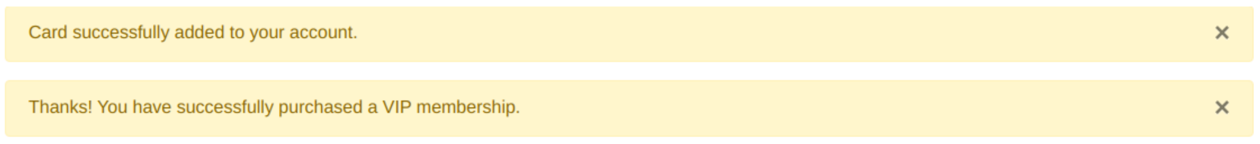
Security code

153

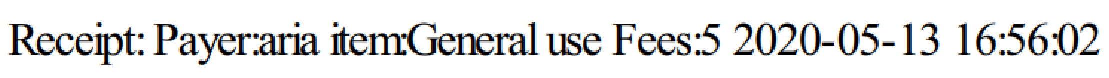
VIP

£45.00

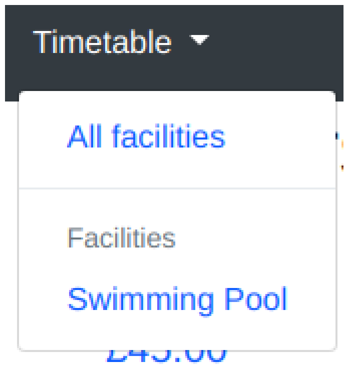
付款成功后会弹出如下提示，并且自动生成 pdf 收据和二维码信息：


Card successfully added to your account.

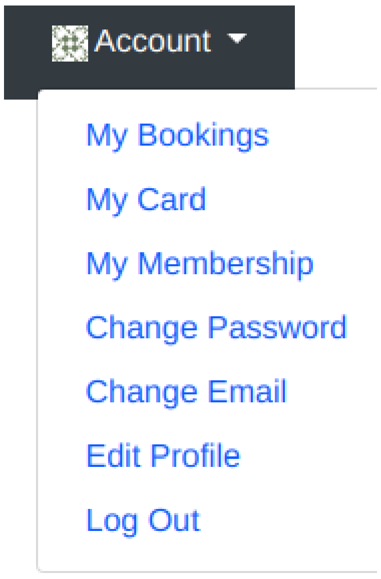
Thanks! You have successfully purchased a VIP membership.


Receipt: Payer:aria item:General use Fees:5 2020-05-13 16:56:02

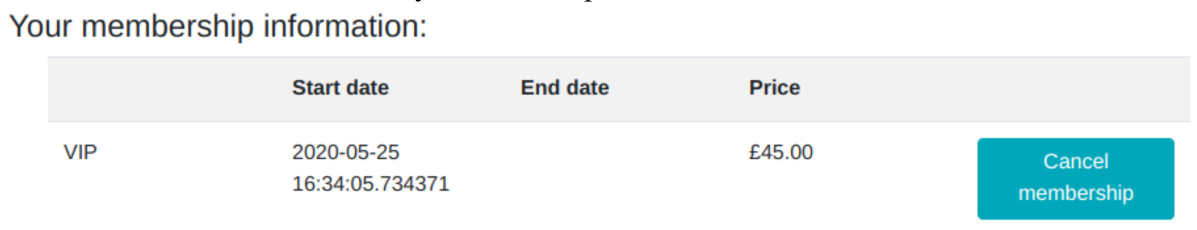
点击菜单栏上方 Pricing List 查看时间表：



点击页面右上方 account 进行账户管理，管理预定、银行卡和会员信息，以及绑定的邮箱，密码，资料，还可以退出登录：

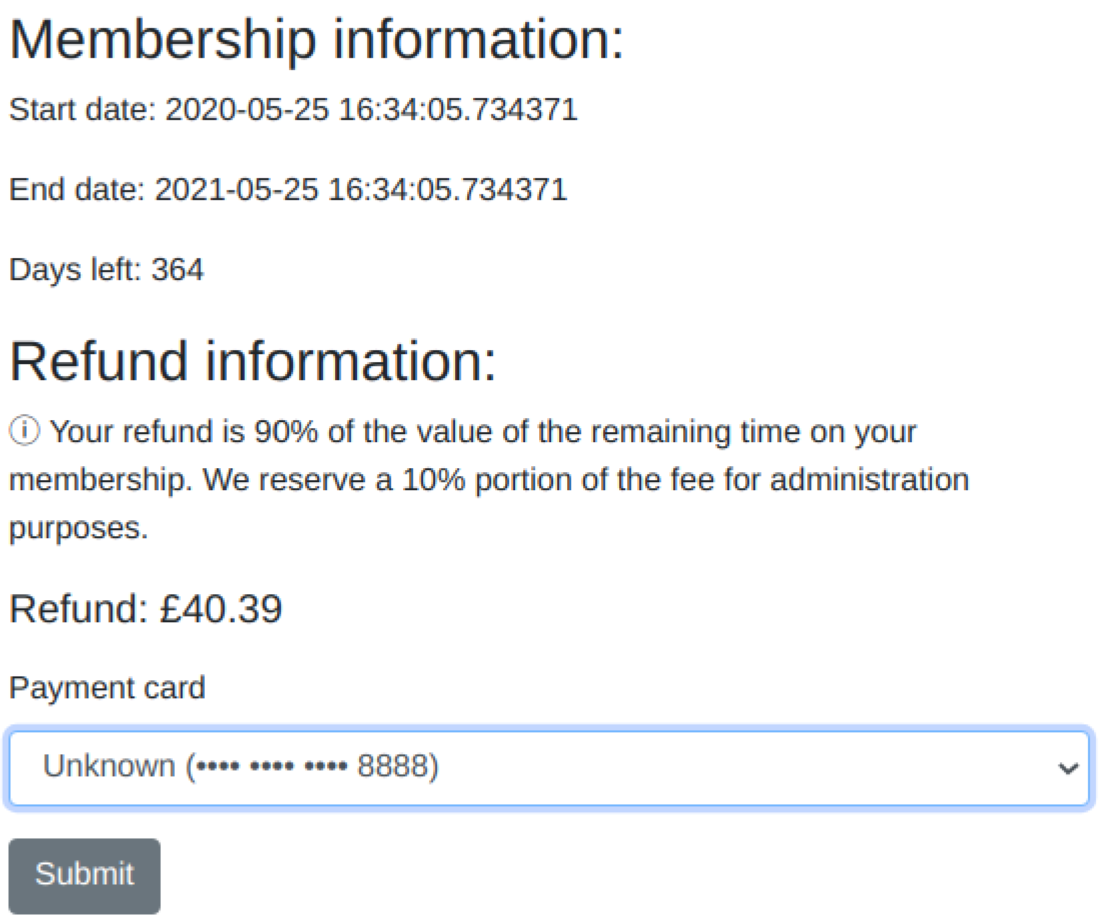


当你需要取消会员时，点击 My Membership，打开以下界面：


Your membership information:

	Start date	End date	Price	
VIP	2020-05-25 16:34:05.734371		£45.00	Cancel membership

点击 Cancel membership，显示退款页面信息，点击 Submit 提交确认：



Membership information:

Start date: 2020-05-25 16:34:05.734371

End date: 2021-05-25 16:34:05.734371

Days left: 364

Refund information:

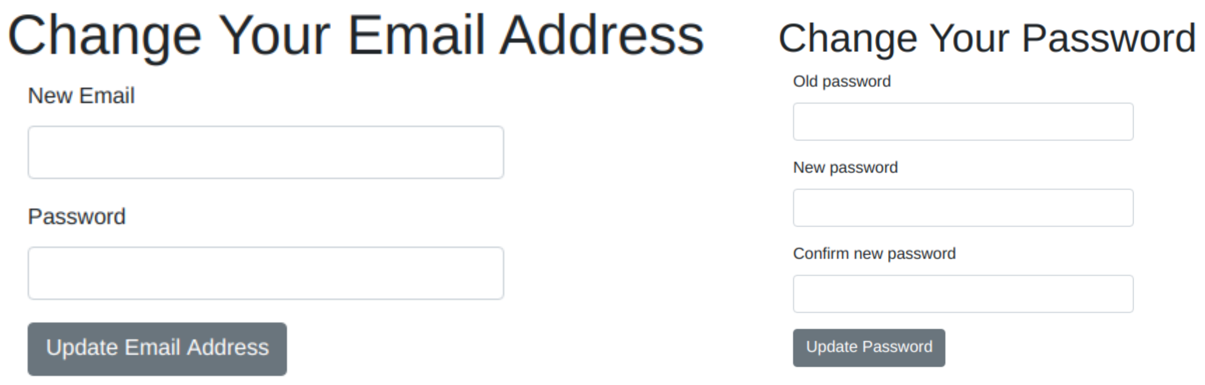
ⓘ Your refund is 90% of the value of the remaining time on your membership. We reserve a 10% portion of the fee for administration purposes.

Refund: £40.39

Payment card

Unknown (•••• •••• •••• 8888)

点击 Change Email 更改邮箱地址，同理点击 Change Password 更改密码：



Change Your Email Address

New Email

Password

Change Your Password

Old password

New password

Confirm new password