

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	<input checked="" type="radio"/> 原创 <input type="radio"/> 修改(含翻译软件、合成软件) <input type="checkbox"/> 修改软件须经原权利人授权 <input type="checkbox"/> 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		<input type="radio"/> 已发表 <input checked="" type="radio"/> 未发表						
开发方式		<input checked="" type="radio"/> 独立开发 <input type="radio"/> 合作开发 <input type="radio"/> 委托开发 <input type="radio"/> 下达任务开发						
著作权人	姓名或名称	类别	证件类型	证件号码	国籍	省份/城市	园区	
	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	<input checked="" type="radio"/> 原始取得 <input type="radio"/> 继受取得 (<input type="radio"/> 受让 <input type="radio"/> 承受 <input type="radio"/> 继承) <input type="checkbox"/> 该软件已登记 (原登记号: __) <input type="checkbox"/> 原登记做过变更或补充 (变更或补充证明编号: __)		
	权利范围	<input checked="" type="radio"/> 全部 <input type="radio"/> 部分 (<input type="checkbox"/> 发表权 <input type="checkbox"/> 署名权 <input type="checkbox"/> 修改权 <input type="checkbox"/> 复制权 <input type="checkbox"/> 发行权 <input type="checkbox"/> 出租权 <input type="checkbox"/> 信息网络传播权 <input type="checkbox"/> 翻译权 <input type="checkbox"/> 应当由著作权人享有的其他权利)		
软件鉴别材料	<input checked="" type="radio"/> 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; <input checked="" type="radio"/> 一种文档 <input type="radio"/> ____种文档		
	<input type="radio"/> 例外交存	<input type="radio"/> 使用黑色宽斜线覆盖, 页码为: <input type="radio"/> 前10页和任选连续的50页 <input type="radio"/> 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		<input checked="" type="radio"/> 由著作权人申请 <input type="radio"/> 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	
<p>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。</p> <p style="text-align: right;">申请人签章：</p> <p style="text-align: right;">2020 年 05 月 25 日</p>				

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。

```
01 # config.py
02
03 import os
04 basedir = os.path.abspath(os.path.dirname(__file__))
05
06 class Config:
07     SECRET_KEY = os.environ.get('SECRET_KEY') or 'hard to guess string'
08
09     MAIL_SERVER = os.environ.get('MAIL_SERVER', '')
10     MAIL_PORT = int(os.environ.get('MAIL_PORT', '587'))
11     MAIL_USE_TLS = os.environ.get('MAIL_USE_TLS', 'true').lower() in \
12         ['true', 'on', '1']
13     MAIL_USERNAME = ''
14     MAIL_PASSWORD = ''
15     FLASKY_MAIL_SUBJECT_PREFIX = '[Flasky]'
16     FLASKY_MAIL_SENDER = ''
17     FLASKY_ADMIN = os.environ.get('FLASKY_ADMIN')
18     SQLALCHEMY_TRACK_MODIFICATIONS = False
19
20     @staticmethod
21     def init_app(app):
22         pass
23
24 class DevelopmentConfig(Config):
25     DEBUG = True
26     SQLALCHEMY_DATABASE_URI = os.environ.get('DEV_DATABASE_URL') or \
27         'sqlite:/// ' + os.path.join(basedir, 'data-dev.sqlite')
28
29 class TestingConfig(Config):
30     TESTING = True
31     SQLALCHEMY_DATABASE_URI = os.environ.get('TEST_DATABASE_URL') or \
32         'sqlite://'
33
34 class ProductionConfig(Config):
35     SQLALCHEMY_DATABASE_URI = os.environ.get('DEV_DATABASE_URL') or \
36         'sqlite:/// ' + os.path.join(basedir, 'data-dev.sqlite')
37     #use ProductionConfig to initialize app
38     @classmethod
39     def init_app(cls, app):
40         Config.init_app(app)
41
42         # email errors to the administrators
43         # setup logger
44         # import logging
45         # from logging.handlers import SMTPHandler
46         # credentials = None
47         # secure = None
```

```
47         # if getattr(cls, 'MAIL_USERNAME', None) is not None:
48         #     credentials = (cls.MAIL_USERNAME, cls.MAIL_PASSWORD)
49         #     if getattr(cls, 'MAIL_USE_TLS', None):
50         #         secure = ()
51         # mail_handler = SMTPHandler(
52         #     mailhost=(cls.MAIL_SERVER, cls.MAIL_PORT),
53         #     fromaddr=cls.FLASKY_MAIL_SENDER,
54         #     toaddrs=[cls.FLASKY_ADMIN],
55         #     subject=cls.FLASKY_MAIL_SUBJECT_PREFIX + ' Application E
rror',
56         #     credentials=credentials,
57         #     secure=secure)
58         # #only send Error message to admin
59         # mail_handler.setLevel(logging.ERROR)
60         # app.logger.addHandler(mail_handler)
61
62         import logging
63         # setup flask log
64         app.debug=True
65         handler = logging.FileHandler('flask.log', encoding='UTF-8')
66         handler.setLevel(logging.INFO)
67         logging_format = logging.Formatter(
68             '%(asctime)s - %(levelname)s - %(filename)s - %(funcName)s
- %(lineno)s - %(message)s')
69         handler.setFormatter(logging_format)
70         app.logger.addHandler(handler)
71
72 config = {
73     'development': DevelopmentConfig,
74     'testing': TestingConfig,
75     'production': ProductionConfig,
76
77     'default': DevelopmentConfig
78 }
79
80 # flasky.py
81
82 import os
83 import click
84 from flask_migrate import Migrate
85 from app import create_app, db
86
87 from app.models import User, Role, Permission, Membership_type, Membership,
Facility, Credit_card_info, Activity, Account, Booking, Time_management
88 app = create_app(os.getenv('FLASK_CONFIG') or 'default')
89 migrate = Migrate(app, db)
90
```

```
91 @app.shell_context_processor
92 def make_shell_context():
93     return dict(db=db, User=User, Role=Role, Permission=Permission, Membership_type=Membership_type, Membership=Membership,
94               Facility=Facility, Booking=Booking, Time_management=Time
95               _management, Activity=Activity, Credit_card_info=Credit_card_info,
96               Account=Account)
97
98 @app.cli.command()
99 @click.argument('test_names', nargs=-1)
100 def test(test_names):
101     """Run the unit tests."""
102     import unittest
103     if test_names:
104         tests = unittest.TestLoader().loadTestsFromNames(test_names)
105     else:
106         tests = unittest.TestLoader().discover('tests')
107     unittest.TextTestRunner(verbosity=2).run(tests)
108
109 # app/__init__.py
110 from flask import Flask
111 from flask_bootstrap import Bootstrap
112 from flask_mail import Mail
113 from flask_moment import Moment
114 from flask_sqlalchemy import SQLAlchemy
115 from flask_login import LoginManager
116 from config import config
117
118 bootstrap = Bootstrap()
119 mail = Mail()
120 moment = Moment()
121 db = SQLAlchemy()
122
123 login_manager = LoginManager()
124 login_manager.login_view = 'auth.login'
125
126 def create_app(config_name):
127     app = Flask(__name__)
128     app.config.from_object(config[config_name])
129     config[config_name].init_app(app)
130
131     bootstrap.init_app(app)
132     mail.init_app(app)
133     moment.init_app(app)
134     db.init_app(app)
135     login_manager.init_app(app)
136
```

```
137     from .main import main as main_blueprint
138     app.register_blueprint(main_blueprint)
139
140     from .auth import auth as auth_blueprint
141     app.register_blueprint(auth_blueprint, url_prefix='/auth')
142
143     return app
144
145 # app/decorators.py
146
147 from functools import wraps
148 from flask import abort
149 from flask_login import current_user
150 from .models import Permission
151
152 def permission_required(permission):
153     def decorator(f):
154         @wraps(f)
155         def decorated_function(*args, **kwargs):
156             if not current_user.can(permission):
157                 abort(403)
158             return f(*args, **kwargs)
159         return decorated_function
160     return decorator
161
162 def admin_required(f):
163     return permission_required(Permission.ADMIN)(f)
164
165 # app/email.py
166
167 from threading import Thread
168 from flask import current_app, render_template
169 from flask_mail import Message
170 from . import mail
171 import pdfkit
172
173 def send_async_email(app, msg):
174     with app.app_context():
175         mail.send(msg)
176
177 def send_email(to, subject, template, **kwargs):
178     app = current_app._get_current_object()
179     msg = Message(app.config['FLASKY_MAIL_SUBJECT_PREFIX'] + ' ' + subject,
180                  sender=app.config['FLASKY_MAIL_SENDER'], recipients=[to])
181     msg.body = render_template(template + '.txt', **kwargs)
```



```
182     msg.html = render_template(template + '.html', **kwargs)
183     thr = Thread(target=send_async_email, args=[app, msg])
184     thr.start()
185     return thr
186
187 def str_to_pdf(string, to_file):
188     # 将 wkhtmltopdf.exe 程序绝对路径传入 config 对象
189     path_wkhtmltopdf = r'C:\\Program Files\\wkhtmltopdf\\bin\\wkhtmltopdf.exe'
190     config = pdfkit.configuration(wkhtmltopdf=path_wkhtmltopdf)
191     # 生成 pdf 文件, to_file 为文件路径
192     pdfkit.from_string(string, to_file, configuration=config)
193     print('完成')
194
195 # app/models.py
196
197 from datetime import datetime, timedelta
198 import hashlib
199
200 from werkzeug.security import generate_password_hash, check_password_hash
201
202 from itsdangerous import TimedJSONWebSignatureSerializer as Serializer
203 from flask import current_app, request
204 from flask_login import UserMixin, AnonymousUserMixin
205 from . import db, login_manager
206
207 class Permission:
208     PAY= 1
209     CANCEL = 2
210     HANDLE = 3
211     DISPLAY = 4
212     VIEW_BUSINESS = 5
213     CONFIGURE = 6
214     MODERATE = 8
215     ADMIN = 16
216
217 class Role(db.Model):
218     __tablename__ = 'roles'
219     id = db.Column(db.Integer, primary_key=True)
220     name = db.Column(db.String(64), unique=True)
221     default = db.Column(db.Boolean, default=False, index=True)
222     permissions = db.Column(db.Integer)
223     users = db.relationship('User', backref='role', lazy='dynamic')
224
225     def __init__(self, **kwargs):
226         super(Role, self).__init__(**kwargs)
```

```
225         if self.permissions is None:
226             self.permissions = 0
227
228     @staticmethod
229     def insert_roles():
230         roles = {
231             'User': [Permission.PAY, Permission.CANCEL, Permission.HA
NDLE,Permission.DISPLAY],
232             'Employee':[Permission.PAY, Permission.CANCEL, Permission
.HANDLE,Permission.DISPLAY],
233             'Manager': [Permission.VIEW_BUSINESS, Permission.CONFIGUR
E],
234
235             'Moderator': [ Permission.MODERATE],
236             'Administrator': [ Permission.MODERATE,Permission.ADMIN],
237
238         }
239         default_role = 'User'
240         for r in roles:
241             role = Role.query.filter_by(name=r).first()
242             if role is None:
243                 role = Role(name=r)
244                 role.reset_permissions()
245                 for perm in roles[r]:
246                     role.add_permission(perm)
247                 role.default = (role.name == default_role)
248                 db.session.add(role)
249             db.session.commit()
250
251     def add_permission(self, perm):
252         if not self.has_permission(perm):
253             self.permissions += perm
254
255     def remove_permission(self, perm):
256         if self.has_permission(perm):
257             self.permissions -= perm
258
259     def reset_permissions(self):
260         self.permissions = 0
261
262     def has_permission(self, perm):
263         return self.permissions & perm == perm
264
265     def __repr__(self):
266         return '<Role %r>' % self.name
267
268 class Membership_type(db.Model):
269     __tablename__ = 'membership_type'
270     id=db.Column(db.Integer,primary_key=True)
271     length=db.Column(db.Integer,nullable=False)
```

```
272     price=db.Column(db.Float,nullable=False)
273     memberships = db.relationship('Membership',backref='user',lazy='dynamic')
274
275 class Membership(db.Model): #have a one-to-
one relationship with user since one user can only have at most one membership
276     __tablename__ = 'membership'
277     id=db.Column(db.Integer,primary_key=True)
278     title = db.Column(db.String(8), index=True)
279     firstname = db.Column(db.String(32), index=True)
280     lastname = db.Column(db.String(32), index=True)
281     status = db.Column(db.String(16), nullable=False)
282     start_date=db.Column(db.DateTime, default=datetime.utcnow)
283     end_date=db.Column(db.DateTime, default=datetime.utcnow)
284     membership_type_id=db.Column(db.Integer,db.ForeignKey('membership
_type.id'))
285     account_id=db.Column(db.Integer,db.ForeignKey('accounts.id'))
286
287     def valueOfEnd_date(self,length):
288         self.end_date = self.start_date + timedelta(days=length)
289         db.session.add(self)
290
291 class User(UserMixin, db.Model): #customers
292     __tablename__ = 'users'
293     id = db.Column(db.Integer, primary_key=True)
294     email = db.Column(db.String(64), unique=True, index=True)
295     username = db.Column(db.String(64), unique=True, index=True)
296     role_id = db.Column(db.Integer, db.ForeignKey('roles.id'),default
=1)
297     password_hash = db.Column(db.String(128))
298     confirmed = db.Column(db.Boolean, default=False)
299
300     def __init__(self, **kwargs):
301         super(User, self).__init__(**kwargs)
302         if self.role is None:
303             if self.email == current_app.config['FLASKY_ADMIN']:
304                 self.role = Role.query.filter_by(name='Administrator'
).first()
305             if self.role is None:
306                 self.role = Role.query.filter_by(default=True).first(
)
307
308     @property
309     def password(self):
310         raise AttributeError('password is not a readable attribute')
311
312     @password.setter
313     def password(self, password):
314         self.password_hash = generate_password_hash(password)
```

```
315
316     def verify_password(self, password):
317         return check_password_hash(self.password_hash, password)
318
319     def generate_confirmation_token(self, expiration=3600):
320         s = Serializer(current_app.config['SECRET_KEY'], expiration)
321         return s.dumps({'confirm': self.id}).decode('utf-8')
322
323     def confirm(self, token):
324         s = Serializer(current_app.config['SECRET_KEY'])
325         try:
326             data = s.loads(token.encode('utf-8'))
327         except:
328             return False
329         if data.get('confirm') != self.id:
330             return False
331         self.confirmed = True
332         db.session.add(self)
333         return True
334
335     def generate_reset_token(self, expiration=3600):
336         s = Serializer(current_app.config['SECRET_KEY'], expiration)
337         return s.dumps({'reset': self.id}).decode('utf-8')
338
339     @staticmethod
340     def reset_password(token, new_password):
341         s = Serializer(current_app.config['SECRET_KEY'])
342         try:
343             data = s.loads(token.encode('utf-8'))
344         except:
345             return False
346         user = User.query.get(data.get('reset'))
347         if user is None:
348             return False
349         user.password = new_password
350         db.session.add(user)
351         return True
352
353     def generate_email_change_token(self, new_email, expiration=3600):
354         s = Serializer(current_app.config['SECRET_KEY'], expiration)
355         return s.dumps(
356             {'change_email': self.id, 'new_email': new_email}).decode(
357             ('utf-8'))
358
359     def change_email(self, token):
360         s = Serializer(current_app.config['SECRET_KEY'])
361         try:
362             data = s.loads(token.encode('utf-8'))
```

```
363         return False
364     if data.get('change_email') != self.id:
365         return False
366     new_email = data.get('new_email')
367     if new_email is None:
368         return False
369     if self.query.filter_by(email=new_email).first() is not None:
370         return False
371     self.email = new_email
372     db.session.add(self)
373     return True
374
375     def can(self, perm):
376         return self.role is not None and self.role.has_permission(per
m)
377
378     def is_administrator(self):
379         return self.can(Permission.ADMIN)
380
381     def ping(self):
382         self.last_seen = datetime.utcnow()
383         db.session.add(self)
384
385     def gravatar(self, size=100, default='identicon', rating='g'):
386         url = 'https://secure.gravatar.com/avatar'
387         hash = hashlib.md5(self.email.lower().encode('utf-
8')).hexdigest()
388         return '{url}/{hash}?s={size}&d={default}&r={rating}'.format(
389             url=url, hash=hash, size=size, default=default, rating=ra
ting)
390
391     def __repr__(self):
392         return '<User %r>' % self.username
393
394 class AnonymousUser(AnonymousUserMixin):
395     def can(self, permissions):
396         return False
397
398     def is_administrator(self):
399         return False
400
401 login_manager.anonymous_user = AnonymousUser
402
403 @login_manager.user_loader
404 def load_user(user_id):
405     return User.query.get(int(user_id))
406
407 """
```

```
408
#each staff can manage many facilities and each facility can be managed b
y many staffs.
409 manage_facility_table=db.Table('manage_facility_table',
410                                db.Column('staff_id', db.Integer, db.F
oreignKey('staff.id'), primary_key=True),
411                                db.Column('facility_id', db.Integer, d
b.ForeignKey('facility.id'), primary_key=True))
412                                """
413
414
class Facility(db.Model): #facility table, each facility can have many co
urts and activities
415     __tablename__ = 'facilities'
416     id=db.Column(db.Integer,primary_key=True)
417     name=db.Column(db.String(16),nullable=False)
418     url = db.Column(db.String(256), nullable=False)
419     capacity=db.Column(db.Integer)
420     description=db.Column(db.String(64),nullable=False)
421     times = db.relationship('Time_management', backref='court',
422                             lazy='dynamic') # each court have many i
ntervals for booking
423     '''courts=db.relationship('Court',backref='facility')'''
424     activities=db.relationship('Activity',backref='facility')
425
426 class Booking(db.Model): #each user can have many bookings
427     __tablename__ = 'bookings'
428     id=db.Column(db.Integer,primary_key=True)
429     number=db.Column(db.Integer)
430     activity=db.Column(db.String(16))
431     activity_id = db.Column(db.Integer,db.ForeignKey('activities.id')
)
432     status=db.Column(db.String(16),nullable=False)
433     payment=db.Column(db.String(32),default='Cash')
434     fees=db.Column(db.Integer,nullable=False)
435     timestamp = db.Column(db.DateTime, default=datetime.utcnow)
436     time_id = db.Column(db.Integer, db.ForeignKey("timetables.id"))
437     account_id=db.Column(db.Integer,db.ForeignKey('accounts.id'))
438
439 class Time_management(db.Model):
440     __tablename__ = 'timetables'
441     id=db.Column(db.Integer,primary_key=True)
442     date=db.Column(db.Date,nullable=False)
443     start_time = db.Column(db.Integer,nullable=False)
444     end_time = db.Column(db.Integer,nullable=False)
445     current_capacity = db.Column(db.Integer,default=0)
446     '''booking_id=db.Column(db.Integer,db.ForeignKey(Booking.id),defa
ult=-1)'''
447     bookings=db.relationship('Booking',backref="timetable")
448     facility_id=db.Column(db.Integer,db.ForeignKey(Facility.id))
```

```
449
450 """
451
class Court(db.Model): #e.g like basketball court, there are 7 courts for
    use
452     __tablename__ = 'courts'
453     id=db.Column(db.Integer,primary_key=True)
454     number=db.Column(db.Integer)
455     availability=db.Column(db.String,default='available')
456     times=db.relationship('Time_management',backref='court',lazy='dynamic') #each court have many intervals for booking
457     facility_id = db.Column(db.Integer, db.ForeignKey('facilities.id'
    ))
458 """
459
460 class Activity(db.Model): #activity
461     __tablename__ = 'activities'
462     id = db.Column(db.Integer, primary_key=True)
463     weekly_income = db.Column(db.Integer)
464     weekly_usage = db.Column(db.Integer)
465     #activity_staff_id=db.Column(db.Integer)
466     activity_price=db.Column(db.Integer)
467     activity_name=db.Column(db.String(16))
468     repeat_every = db.Column(db.Integer)
469     bookings = db.relationship('Booking', backref="activity_ii")
470     facility_id=db.Column(db.Integer,db.ForeignKey('facilities.id'))
471
472 class Credit_card_info(db.Model): #paying card information
473     __tablename__ = 'cards'
474     id=db.Column(db.Integer,primary_key=True)
475     card_number=db.Column(db.Integer,nullable=False)
476     expire_month=db.Column(db.Integer,nullable=False)
477     expire_year=db.Column(db.Integer,nullable=False)
478     security_code=db.Column(db.Integer,nullable=False)
479     holder_name=db.Column(db.String(64),nullable=False)
480     account_id = db.Column(db.Integer, db.ForeignKey('accounts.id'))
481
482 class Account(db.Model):
483     __tablename__='accounts'
484     id=db.Column(db.Integer, primary_key=True, nullable=False)
485     user_id=db.Column(db.Integer,db.ForeignKey('users.id'), unique=True)
486     membership = db.relationship('Membership', uselist=False)
487     cards = db.relationship('Credit_card_info', backref='account', lazy='dynamic')
488     bookings = db.relationship('Booking', backref='account', lazy='dynamic')
489
490 """
491 class Staff(db.Model):
```

```
492     __tablename__ = 'staffs'
493     id=db.Column(db.Integer,primary_key=True)
494     name=db.Column(db.String(64),nullable=False)
495     role=db.Column(db.String(64),nullable=False)
496     facilities = db.relationship('Facility',secondary=manage_facility
__table,backref='Staff')
497     """
498
499 """
500 class Business(db.Model):
501     __tablename__ = 'business'
502     id = db.Column(db.Integer,primart_key=True)
503     people = db.Column(db.Integer)
504     income = db.Column(db.Integer)
505 """
506
507 # app/auth/__init__.py
508
509 from flask import Blueprint
510 auth = Blueprint('auth', __name__)
511 from . import views
512
513 # app/auth/views.py
514
515 from flask import render_template, redirect, request, url_for, flash
516 from flask_login import login_user, logout_user, login_required, \
517     current_user
518 from . import auth
519 from .. import db
520 from ..models import User, Account
521 from ..email import send_email
522 from .forms import LoginForm, RegistrationForm, ChangePasswordForm,\
523     PasswordResetRequestForm, PasswordResetForm, ChangeEmailForm
524
525 @auth.before_app_request
526 def before_request():
527     if current_user.is_authenticated:
528         current_user.ping()
529         if not current_user.confirmed \
530             and request.endpoint \
531             and request.blueprint != 'auth' \
532             and request.endpoint != 'static':
533             return redirect(url_for('auth.unconfirmed'))
534
535 @auth.route('/unconfirmed')
536 def unconfirmed():
537     if current_user.is_anonymous or current_user.confirmed:
538         return redirect(url_for('main.index'))
```



```
539     return render_template('auth/unconfirmed.html')
540
541 @auth.route('/login', methods=['GET', 'POST'])
542 def login():
543     form = LoginForm()
544     if form.validate_on_submit():
545         user = User.query.filter_by(email=form.email.data.lower()).fi
rst()
546         if user is not None and user.verify_password(form.password.da
ta):
547             login_user(user, form.remember_me.data)
548             next = request.args.get('next')
549             if next is None or not next.startswith('/'):
550                 next = url_for('main.index')
551             return redirect(next)
552         flash('Invalid email or password.')
553     return render_template('auth/login.html', form=form)
554
555 @auth.route('/logout')
556 @login_required
557 def logout():
558     logout_user()
559     flash('You have been logged out.')
560     return redirect(url_for('main.index'))
561
562 @auth.route('/register', methods=['GET', 'POST'])
563 def register():
564     form = RegistrationForm()
565     if form.validate_on_submit():
566         user = User(email=form.email.data.lower(),
567                     username=form.username.data,
568                     password=form.password.data)
569         db.session.add(user)
570         db.session.commit()
571         token = user.generate_confirmation_token()
572         send_email(user.email, 'Confirm Your Account',
573                 'auth/email/confirm', user=user, token=token)
574         flash('A confirmation email has been sent to you by email.')
575         return redirect(url_for('auth.login'))
576     return render_template('auth/register.html', form=form)
577
578 @auth.route('/confirm/<token>')
579 @login_required
580 def confirm(token):
581     if current_user.confirmed:
582         return redirect(url_for('main.index'))
583     if current_user.confirm(token):
```

```
584         db.session.commit()
585         account = Account(user_id=current_user.id)
586         db.session.add(account)
587         db.session.commit()
588         flash('You have confirmed your account. Thanks!')
589     else:
590         flash('The confirmation link is invalid or has expired.')
591     return redirect(url_for('main.index'))
592
593 @auth.route('/confirm')
594 @login_required
595 def resend_confirmation():
596     token = current_user.generate_confirmation_token()
597     send_email(current_user.email, 'Confirm Your Account',
598               'auth/email/confirm', user=current_user, token=token)
599     flash('A new confirmation email has been sent to you by email.')
600     return redirect(url_for('main.index'))
601
602 @auth.route('/change-password', methods=['GET', 'POST'])
603 @login_required
604 def change_password():
605     form = ChangePasswordForm()
606     if form.validate_on_submit():
607         if current_user.verify_password(form.old_password.data):
608             current_user.password = form.password.data
609             db.session.add(current_user)
610             db.session.commit()
611             flash('Your password has been updated.')
612             return redirect(url_for('main.index'))
613         else:
614             flash('Invalid password.')
615     return render_template("auth/change_password.html", form=form)
616
617 @auth.route('/reset', methods=['GET', 'POST'])
618 def password_reset_request():
619     if not current_user.is_anonymous:
620         return redirect(url_for('main.index'))
621     form = PasswordResetRequestForm()
622     if form.validate_on_submit():
623         user = User.query.filter_by(email=form.email.data.lower()).fi
rst()
624         if user:
625             token = user.generate_reset_token()
626             send_email(user.email, 'Reset Your Password',
627                       'auth/email/reset_password',
628                       user=user, token=token)
```

```
629         flash('An email with instructions to reset your password has
been '
630             'sent to you.')
631         return redirect(url_for('auth.login'))
632     return render_template('auth/reset_password.html', form=form)
633
634 @auth.route('/reset/<token>', methods=['GET', 'POST'])
635 def password_reset(token):
636     if not current_user.is_anonymous:
637         return redirect(url_for('main.index'))
638     form = PasswordResetForm()
639     if form.validate_on_submit():
640         if User.reset_password(token, form.password.data):
641             db.session.commit()
642             flash('Your password has been updated.')
643             return redirect(url_for('auth.login'))
644         else:
645             return redirect(url_for('main.index'))
646     return render_template('auth/reset_password.html', form=form)
647
648 @auth.route('/change_email', methods=['GET', 'POST'])
649 @login_required
650 def change_email_request():
651     form = ChangeEmailForm()
652     if form.validate_on_submit():
653         if current_user.verify_password(form.password.data):
654             new_email = form.email.data.lower()
655             token = current_user.generate_email_change_token(new_email)
656             send_email(new_email, 'Confirm your email address',
657                 'auth/email/change_email',
658                 user=current_user, token=token)
659             flash('An email with instructions to confirm your new email
il '
660                 'address has been sent to you.')
661             return redirect(url_for('main.index'))
662         else:
663             flash('Invalid email or password.')
664     return render_template("auth/change_email.html", form=form)
665
666 @auth.route('/change_email/<token>')
667 @login_required
668 def change_email(token):
669     if current_user.change_email(token):
670         db.session.commit()
671         flash('Your email address has been updated.')
672     else:
```

```
673         flash('Invalid request.')
674         return redirect(url_for('main.index'))
675
676 # app/auth/forms.py
677
678 from flask_wtf import FlaskForm
679
680 from wtforms import StringField, PasswordField, BooleanField, SubmitField
681 , SelectField
682
683 from wtforms.validators import DataRequired, Length, Email, Regexp, Equal
684 To
685
686 from wtforms import ValidationError
687
688 from ..models import User
689
690
691 class LoginForm(FlaskForm):
692     email = StringField('Email', validators=[DataRequired(), Length(1
693 , 64),
694                                     Email()]])
695     password = PasswordField('Password', validators=[DataRequired()])
696     remember_me = BooleanField('Keep me logged in')
697     submit = SubmitField('Log In')
698
699
700 class RegistrationForm(FlaskForm):
701     email = StringField('Email', validators=[DataRequired(), Length(1
702 , 64),
703                                     Email()]])
704     username = StringField('Username', validators=[
705         DataRequired(), Length(1, 64),
706         Regexp('^[A-Za-z][A-Za-z0-9_]*$', 0,
707             'Usernames must have only letters, numbers, dots or '
708             'underscores')])
709     role = SelectField('Role', choices=[('Customer', 'Customer'),
710 ('Employee', 'Employee'), ('Manager', 'Manager')])
711     password = PasswordField('Password', validators=[
712         DataRequired(), EqualTo('password2', message='Passwords must
713 match.')]])
714     password2 = PasswordField('Confirm password', validators=[DataReq
715 uired()])
716     submit = SubmitField('Register')
717
718     def validate_email(self, field):
719         if User.query.filter_by(email=field.data.lower()).first():
720             raise ValidationError('Email already registered.')
721
722     def validate_username(self, field):
723         if User.query.filter_by(username=field.data).first():
724             raise ValidationError('Username already in use.')
```

712

713 class ChangePasswordForm(FlaskForm):

714 old_password = PasswordField('Old password', validators=[DataRequired()])

715 password = PasswordField('New password', validators=[
716 DataRequired(), EqualTo('password2', message='Passwords must
match.')]])

717 password2 = PasswordField('Confirm new password',
718 validators=[DataRequired()])

719 submit = SubmitField('Update Password')

720

721 class PasswordResetRequestForm(FlaskForm):

722 email = StringField('Email', validators=[DataRequired(), Length(1
723 , 64),

Email()]])

724 submit = SubmitField('Reset Password')

725

726 class PasswordResetForm(FlaskForm):

727 password = PasswordField('New Password', validators=[
728 DataRequired(), EqualTo('password2', message='Passwords must
match')]])

729 password2 = PasswordField('Confirm password', validators=[DataRequired()])

730 submit = SubmitField('Reset Password')

731

732 class ChangeEmailForm(FlaskForm):

733 email = StringField('New Email', validators=[DataRequired(), Length(1, 64),

Email()]])

735 password = PasswordField('Password', validators=[DataRequired()])

736 submit = SubmitField('Update Email Address')

737

738 def validate_email(self, field):

739 if User.query.filter_by(email=field.data.lower()).first():

740 raise ValidationError('Email already registered.')

741

742 # app/main/__init__.py

743

744 from flask import Blueprint

745

746 main = Blueprint('main', __name__)

747

748 from . import views, errors

749 from ..models import Permission

750

751 @main.app_context_processor

```
752 def inject_permissions():
753     return dict(Permission=Permission)
754
755 # app/main/error.py
756
757 from flask import render_template
758 from . import main
759
760 @main.app_errorhandler(403)
761 def forbidden(e):
762     return render_template('403.html'), 403
763
764 @main.app_errorhandler(404)
765 def page_not_found(e):
766     return render_template('404.html'), 404
767
768 @main.app_errorhandler(500)
769 def internal_server_error(e):
770     return render_template('500.html'), 500
771
772 # app/main/views.py
773
774 import datetime
775
776
777 from flask import render_template, redirect, url_for, abort, flash, request, current_app
778 from flask_login import login_required, current_user
779 from . import main
780 from .forms import EditProfileForm, EditProfileAdminForm, MembershipForm, PoolBookingForm, \
781     FitnessBookingForm, SquashBookingForm, HallBookingForm, CardForm, RefundForm, ConfigureActivityForm, \
782     ConfigureTimetableForm, \
783     ConfigureFacilityForm, SearchForm
784 from .. import db
785 from ..models import Role, User, Facility, Membership, Account, Activity, Time_management, Booking, Credit_card_info, \
786     Membership_type, Permission
787 from ..decorators import admin_required, permission_required
788 from ..email import send_email, str_to_pdf
789 import qrcode
790 import pdfkit
791
792 @main.route('/')
793
```

```
792 def index():
793     return render_template('index.html')
794
795 @main.route('/facility')
796 def facility():
797     facilities = Facility.query.all()
798     '''for facility in facilities:
799         print(facility.name)'''
800     return render_template('facility.html', facilities=facilities)
801
802 @main.route('/facility_info/<int:id>')
803 def view_info(id):
804     facility = Facility.query.filter_by(id=id).first()
805     return render_template('facility_info.html', facility=facility)
806
807 @main.route('/membership')
808 def view_membership_type():
809     memberships = Membership_type.query.all()
810     for membership in memberships:
811         print(membership.length)
812     return render_template('membership.html', memberships=memberships)
813
814 @main.route('/my_bookings/<int:id>')
815 @login_required
816 @permission_required(Permission.DISPLAY)
817 def display_bookings(id):
818     account = Account.query.filter_by(user_id=id).first()
819     bookings = Booking.query.order_by(Booking.timestamp.desc()).filter_by(account_id=account.id).all()
820     number = Booking.query.order_by(Booking.timestamp.desc()).filter_by(account_id=account.id).count()
821     facilities = Facility.query.all()
822     timetable_booking = []
823     for booking in bookings:
824         timetable = Time_management.query.filter_by(id=booking.time_id).first()
825         timetable_booking.append(timetable)
826
827     return render_template('my_booking.html', number=number, bookings=bookings, timetable_booking=timetable_booking, facilities=facilities)
828
829
830 @main.route('/my_card/<int:id>')
831 @login_required
832 @permission_required(Permission.DISPLAY)
```

```
833 def display_cards(id):
834     account = Account.query.filter_by(user_id=id).first()
835     cards = Credit_card_info.query.filter_by(account_id=account.id).all()
836     number = Credit_card_info.query.filter_by(account_id=account.id).count()
837     return render_template('my_card.html', cards=cards, number=number)
838
839 @main.route('/remove_card/<int:id>', methods=['GET', 'POST'])
840 @login_required
841 @permission_required(Permission.DISPLAY)
842 def remove_card(id):
843     card = Credit_card_info.query.get_or_404(id)
844     db.session.delete(card)
845     db.session.commit()
846     flash('Successfully remove a card.')
847     # The function will go to all function and render the template in dex.html
848     return redirect(url_for('.display_cards', id=current_user.id))
849
850 @main.route('/my_membership/<int:id>')
851 @login_required
852 @permission_required(Permission.DISPLAY)
853 def display_membership(id):
854     account = Account.query.filter_by(user_id=id).first()
855     membership = Membership.query.filter_by(account_id=account.id).first()
856     membership_type = Membership_type.query.filter_by(id=membership.membership_type_id).first()
857     number = Membership.query.filter_by(account_id=account.id).count()
858     return render_template('my_membership.html', membership=membership, membership_type=membership_type, number=number)
859
860 @main.route('/pricing_list')
861 def display_pricing_list():
862     facilities = Facility.query.all()
863     activities = Activity.query.all()
864     return render_template('pricing_list.html', facilities=facilities, activities=activities)
865
866 @main.route('/timetable_all/<int:date>/<int:offset>')
867 def timetable_all(date, offset):
868     if date == 2 and offset == 0:
869         date = 0
870     if date == 2:
```



```
871         date = -1
872         newdate = offset + date
873         new_date = datetime.date.today() + datetime.timedelta(days=newdate)
874         offset += date
875
876         timetables = Time_management.query.filter_by(date=new_date).all()
877         facilities = Facility.query.all()
878
879         return render_template('timetable_all.html', offset=offset, timetables=timetables, facilities=facilities,
880                                new_date=new_date)
881
882 @main.route('/timetable_facility/<int:type>')
883 def timetable_facility(type):
884     if type == 1:
885         type = type + 1
886         facility = "Swimming pool"
887     elif type == 2:
888         type = type + 1
889         facility = "Fitness room"
890     elif type == 3:
891         type = type + 1
892         facility = "Squash courts"
893     else:
894         type = 1
895         facility = "Sports hall"
896
897     facility = Facility.query.filter_by(name=facility).first()
898     today = datetime.date.today()
899     day2 = today + datetime.timedelta(days=1)
900     day3 = today + datetime.timedelta(days=2)
901     day4 = today + datetime.timedelta(days=3)
902     day5 = today + datetime.timedelta(days=4)
903     timetable1 = Time_management.query.filter_by(date=today).all()
904     timetable2 = Time_management.query.filter_by(date=day2).all()
905     timetable3 = Time_management.query.filter_by(date=day3).all()
906     timetable4 = Time_management.query.filter_by(date=day4).all()
907     timetable5 = Time_management.query.filter_by(date=day5).all()
908     return render_template('timetable_facility.html', type=type, today=today, day3=day3, day2=day2, day4=day4,
909                            day5=day5, facility=facility, timetable1=timetable1, timetable2=timetable2,
910                            timetable3=timetable3, timetable4=timetable4, timetable5=timetable5)
911
912 @main.route('/book_facility/<int:type>', methods=['GET', 'POST'])
913 @login_required
914 @permission_required(Permission.PAY)
```

```
915 def book_facility(type):
916     if type == 1:
917         form = PoolBookingForm()
918     elif type == 2:
919         form = FitnessBookingForm()
920     elif type == 3:
921         form = SquashBookingForm()
922     else:
923         form = HallBookingForm()
924
925     if form.validate_on_submit():
926         '''activity = form.activity.data
927         number = form.number.data
928         date = form.date.data
929         start_time = form.start_time.data
930         end_time = form.end_time.data
931         payment = form.payment.data'''
932
933         fac = Facility.query.filter_by(id=type).first()
934         account = Account.query.filter_by(user_id=current_user.id).fi
rst()
935         for booking in account.bookings:
936             book_timetable = Time_management.query.filter_by(id=booki
ng.time_id).first()
937             if book_timetable.date == form.date.data:
938                 flash("You can only have exactly one booking each day
")
939                 return redirect(url_for('.book_facility', type=type))
940
941         if form.end_time.data - form.start_time.data != 1:
942             flash('You can only book a 1-hour session')
943             return redirect(url_for('.book_facility', type=type))
944
945         timetables = Time_management.query.filter_by(facility_id=type
).all()
946         for timetable in timetables:
947             if form.start_time.data == timetable.start_time and form.
date.data ==timetable.date:
948                 time_id = timetable.id
949
950             timetable_booking = Time_management.query.filter_by(id=time_i
d).first()
951             if form.number.data + timetable_booking.current_capacity > fa
c.capacity:
952                 flash('Your have too many people!')
953                 return redirect(url_for('.book_facility', type=type))
954
955         else:
956             account = Account.query.filter_by(user_id=current_user.id
).first()
```

```
957         for activity in fac.activities:
958             print(activity.activity_name)
959             if activity.activity_name == form.activity.data:
960                 fees=activity.activity_price
961                 activity_id=activity.id
962
963             booking = Booking(number=form.number.data,
964                               time_id=time_id,
965                               account_id=account.id,
966                               activity=form.activity.data,
967                               activity_id=activity_id,
968                               status="Unpaid",
969                               payment=form.payment.data,
970                               fees=fees)
971             db.session.add(booking)
972             db.session.commit()
973
974             book = Booking.query.order_by(Booking.timestamp.desc()).f
975             ilter_by(account_id=account.id).first_or_404()
976
977             timetable_booking.current_capacity = form.number.data + t
978             imetable_booking.current_capacity
979             db.session.add(timetable_booking)
980             db.session.commit()
981
982             if form.payment.data == "Credit Card":
983                 flash('Pay for your booking.')
984                 return redirect(url_for('.handle_card_booking', book_
985 id=book.id))
986             elif form.payment.data == "Cash":
987                 flash('You will pay for your booking by cash.')
988                 return redirect(url_for('.index'))
989
990         return render_template('book_facility.html', form=form)
991
992 @main.route('/handle_card_booking/<int:book_id>', methods=['GET', 'POST']
993 )
994 @login_required
995 @permission_required(Permission.HANDLE)
996 def handle_card_booking(book_id):
997     account = Account.query.filter_by(user_id=current_user.id).first(
998 )
999     card = Credit_card_info.query.filter_by(account_id=account.id).fi
1000 rst()
1001     book = Booking.query.filter_by(id=book_id).first()
1002     time = Time_management.query.filter_by(id=book.time_id).first()
1003     form = CardForm()
1004
1005     if form.validate_on_submit():
```

```
1000         length = len(str(form.card_number.data))
1001         length_year = len(str(form.expire_year.data))
1002         length_code = len(str(form.security_code.data))
1003
1004         if card is None or card.card_number != form.card_number.data
:
1005             if length != 16:
1006                 flash("Your card number should have 16 numbers.")
1007                 return redirect(url_for(".handle_card_booking", book
_id=book_id))
1008             if form.expire_month.data > 12 or form.expire_month.data
< 1:
1009                 flash("Your expire month is invalid.")
1010                 return redirect(url_for(".handle_card_booking", book
_id=book_id))
1011             if length_year != 4:
1012                 flash("Your expire year is invalid.")
1013                 return redirect(url_for(".handle_card_booking", book
_id=book_id))
1014             if length_code != 3:
1015                 flash("Your security code is invalid.")
1016                 return redirect(url_for(".handle_card_booking", book
_id=book_id))
1017
1018         card_info = Credit_card_info(card_number=form.card_numbe
r.data,
1019                                     expire_month=form.expire_mo
nth.data,
1020                                     expire_year=form.expire_yea
r.data,
1021                                     security_code=form.security
_code.data,
1022                                     holder_name=form.holder_nam
e.data,
1023                                     account_id=account.id
1024                                     )
1025         db.session.add(card_info)
1026         db.session.commit()
1027         book.status = "Paid"
1028         db.session.add(book)
1029         db.session.commit()
1030
1031         act = Activity.query.filter_by(id=book.activity_id).first()
1032         act.weekly_income = act.weekly_income + book.fees
1033         act.weekly_usage = act.weekly_usage + book.number
1034         db.session.add(act)
1035         db.session.commit()
1036
1037         receipt = "Receipt:\n" + "Payer:" + current_user.username +
"\n" + "item:" + book.activity + "\n" + "Fees:" + str(
```

```
1038         book.fees) + "\n" + book.timestamp.strftime('%Y-%m-%d %H
1039         :%M:%S') + "\n"
1040         pdf_name = 'app/static/' + str(book.id) + '.pdf'
1041         str_to_pdf(receipt, pdf_name)
1042         img = qrcode.make(receipt)
1043         img.save("app/static/receipt.jpg")
1044         send_email(current_user.email, 'Your booking receipt',
1045                 'booking_receipt', book=book, time=time)
1046
1047         flash('A booking receipt has been sent to you by email.')
1048         return redirect(url_for("main.index"))
1049     if card is not None:
1050         form.card_number.data = card.card_number
1051         form.expire_month.data = card.expire_month
1052         form.expire_year.data = card.expire_year
1053         form.security_code.data = card.security_code
1054         form.holder_name.data = card.holder_name
1055     return render_template('handle_card_booking.html', book=book, ti
1056     me=time, form=form)

1057 @main.route('/cancel_booking/<int:id>', methods=['GET', 'POST'])
1058 @login_required
1059 @permission_required(Permission.CANCEL)
1060 def cancel_booking(id):
1061     booking = Booking.query.filter_by(id=id).first()
1062     time = Time_management.query.filter_by(id=booking.time_id).first
1063     ()
1064     if booking.status == "Unpaid":
1065         db.session.delete(booking)
1066         db.session.commit()
1067         return redirect(url_for('main.display_bookings', id=current_
1068         user.id))
1069     else:
1070         form = RefundForm()
1071         if form.validate_on_submit():
1072             time.current_capacity = time.current_capacity - booking.
1073             number
1074             db.session.add(time)
1075             db.session.commit()
1076             db.session.delete(booking)
1077             db.session.commit()
1078             activity = Activity.query.filter_by(activity_name=bookin
1079             g.activity).first()
1080             activity.weekly_usage = activity.weekly_usage - booking.
1081             number
1082             activity.weekly_income = activity.weekly_income - bookin
1083             g.fees
1084             db.session.add(activity)
```

```
1079         db.session.commit()
1080         return redirect(url_for('main.display_bookings', id=current_user.id))
1081
1082     return render_template('cancel_booking.html', form=form, booking=booking, time=time)
1083
1084
1085 @main.route('/pay_membership/<int:type_id>', methods=['GET', 'POST'])
1086 @login_required
1087 def pay_membership(type_id):
1088     membership_type = Membership_type.query.filter_by(id=type_id).first()
1089     account = Account.query.filter_by(user_id=current_user.id).first()
1090     form = MembershipForm()
1091     if form.validate_on_submit():
1092         membership = Membership(title=form.title.data,
1093                                 status="Unpaid",
1094                                 firstname=form.firstname.data,
1095                                 lastname=form.lastname.data,
1096                                 account_id=account.id,
1097                                 membership_type_id=membership_type.id)
1098         db.session.add(membership)
1099         db.session.commit()
1100
1101         membership = Membership.query.filter_by(account_id=account.id).first()
1102         if (membership_type.length == 1):
1103             length = 30
1104         elif (membership_type.length == 3):
1105             length = 90
1106         else:
1107             length = 365
1108         membership.valueOfEnd_date(length)
1109         db.session.commit()
1110
1111         if form.payment.data == "Credit Card":
1112             flash('Pay for your booking.')
1113             return redirect(url_for('.handle_card_membership', id=account.id))
1114         elif form.payment.data == "Cash":
1115             flash('You will pay for your booking by cash.')
1116             return redirect(url_for('.index'))
1117
1118     return render_template('pay_membership.html', form=form, membership_type=membership_type)
```

```
1119
1120
1121 @main.route('/handle_card_membership/<int:id>', methods=['GET', 'POST'])
1122 @login_required
1123 def handle_card_membership(id):
1124     membership = Membership.query.filter_by(account_id=id).first()
1125     card = Credit_card_info.query.filter_by(account_id=id).first()
1126     form = CardForm()
1127     membership_type = Membership_type.query.filter_by(id=membership.
membership_type_id).first()
1128     price = membership_type.price
1129     if form.validate_on_submit():
1130         card_info = Credit_card_info(card_number=form.card_number.da
ta,
1131                                     expire_month=form.expire_month.
data,
1132                                     expire_year=form.expire_year.da
ta,
1133                                     security_code=form.security_cod
e.data,
1134                                     holder_name=form.holder_name.da
ta,
1135                                     account_id=id
)
1136         db.session.add(card_info)
1137         db.session.commit()
1138         membership.status = "Paid"
1139         db.session.add(membership)
1140         db.session.commit()
1141         return redirect(url_for('main.index'))
1142
1143     if card is not None:
1144         form.card_number.data = card.card_number
1145         form.expire_month.data = card.expire_month
1146         form.expire_year.data = card.expire_year
1147         form.security_code.data = card.security_code
1148         form.holder_name.data = card.holder_name
1149
1150     return render_template('handle_card_membership.html', price=pric
e, form=form)
1151
1152 @main.route('/configure_facility', methods=['GET', 'POST'])
1153 @login_required
1154 def configure_facility():
1155     form = ConfigureFacilityForm()
1156     if form.validate_on_submit():
1157         operation = form.operation.data
1158         capacity = form.capacity.data
```

```
1159     name = form.name.data
1160     url = form.url.data
1161     description = form.description.data
1162
1163     if operation == "add":
1164         facility = Facility(name=name,
1165                             url=url,
1166                             capacity=capacity,
1167                             description=description)
1168         db.session.add(facility)
1169         db.session.commit()
1170         flash('You have added the facility')
1171         return redirect(url_for('main.index'))
1172
1173     elif operation == "edit":
1174         fac = Facility.query.filter_by(name=name).first()
1175         fac.capacity = capacity
1176         fac.url = url
1177         fac.description = description
1178
1179         db.session.add(fac)
1180         db.session.commit()
1181         flash('You have edited the facility')
1182         return redirect(url_for('main.index'))
1183
1184     elif operation == "delete":
1185         fac = Facility.query.filter_by(name=name).first()
1186         db.session.delete(fac)
1187         db.session.commit()
1188         flash('You have deleted the facility')
1189         return redirect(url_for('main.index'))
1190
1191     return render_template('configure_facility.html', form=form)
1192
1193 @main.route('/configure_activity', methods=['GET', 'POST'])
1194 @login_required
1195 def configure_activity():
1196     form = ConfigureActivityForm()
1197     if form.validate_on_submit():
1198
1199         fac = Facility.query.filter_by(name=form.facility.data).first()
1200
1201         if form.operation.data == "add":
1202             act = Activity(weekly_income=0,
1203                           weekly_usage=0,
1204                           activity_price=form.price.data,
1205                           activity_name=form.activity.data,
1206                           facility_id=fac.id
1207                           )
```



```
1207         db.session.add(act)
1208         db.session.commit()
1209         flash('You have added the activity')
1210         return redirect(url_for('main.index'))
1211
1212     elif form.operation.data == "edit":
1213         act = Activity.query.filter_by(activity_name=form.activity.data).first()
1214         act.activity_price = form.price.data,
1215         act.activity_name = form.activity.data
1216         act.facility_id = fac.id
1217
1218         db.session.add(act)
1219         db.session.commit()
1220         flash('You have edited the activity')
1221         return redirect(url_for('main.index'))
1222
1223     elif form.operation.data == "delete":
1224         act = Activity.query.filter_by(activity_name=form.activity.data).first()
1225         db.session.delete(act)
1226         db.session.commit()
1227         flash('You have deleted the activity')
1228         return redirect(url_for('main.index'))
1229
1230     return render_template('configure_activity.html', form=form)
1231
1232 @main.route('/configure_timetable', methods=['GET', 'POST'])
1233 @login_required
1234 def configure_timetable():
1235     form = ConfigureTimetableForm()
1236     if form.validate_on_submit():
1237         fac = Facility.query.filter_by(name=form.facility.data).first()
1238
1239         if fac is None:
1240             flash("The facility is not existed")
1241             return redirect(url_for("main.configure_timetable"))
1242
1243         for i in range(form.start_time.data, form.end_time.data):
1244             timetable = Time_management(date=form.date.data,
1245                                         start_time=i,
1246                                         end_time=i + 1,
1247                                         facility_id=fac.id)
1248             db.session.add(timetable)
1249             db.session.commit()
1250             flash("Sucessful operation!")
1251             return redirect(url_for('main.index'))
1252     return render_template('configure_timetable.html', form=form)
```

1253

```
1254 @main.route('/cancel_membership/<int:id>', methods=['GET', 'POST'])
1255 @login_required
1256 @permission_required(Permission.CANCEL)
1257 def cancel_membership(id):
1258     membership = Membership.query.filter_by(id=id).first()
1259     timedelta = membership.end_date - datetime.datetime.utcnow()
1260     days_left = timedelta.days
1261     money_refund = timedelta.days * 0.5
1262     form = RefundForm()
1263     if form.validate_on_submit():
1264         db.session.delete(membership)
1265         db.session.commit()
1266         flash('You have cancelled our membership!')
1267         return redirect(url_for('main.display_membership', id=current_user.id))
1268     return render_template('cancel_membership.html', money_refund=money_refund, days_left=days_left,
1269                             membership=membership, form=form)
1270
```

```
1271 @main.route('/user/<username>')
1272 def user(username):
1273     user = User.query.filter_by(username=username).first_or_404()
1274     return render_template('user.html', user=user)
1275
```

```
1276 @main.route('/search_booking', methods=['GET', 'POST'])
1277 @login_required
1278 @permission_required(Permission.CANCEL)
1279 def search_booking():
1280     form = SearchForm()
1281     if form.validate_on_submit():
1282         email = form.email.data
1283
1284         user = User.query.filter_by(email=email).first()
1285         return redirect(url_for('main.display_bookings', id=user.id))
1286     return render_template('search_booking.html', form=form)
1287
```

```
1288 @main.route('/search_membership', methods=['GET', 'POST'])
1289 @login_required
1290 @permission_required(Permission.CANCEL)
1291 def search_membership():
1292     form = SearchForm()
1293     if form.validate_on_submit():
1294         email = form.email.data
1295
1296         user = User.query.filter_by(email=email).first()
```

```
1297         return redirect(url_for('main.display_membership', id=user.i
1298         return render_template('search_membership.html', form=form)
1299
1300 @main.route('/view_income')
1301 @login_required
1302 def view_income():
1303     session1_income, session2_income, session3_income, session4_inco
1304     me, session5_income, session6_income, session14_income = 0, 0, 0, 0, 0, 0
1305     , 0
1306     session7_income, session8_income, session9_income, session10_inc
1307     ome, session11_income, session12_income, session13_income = 0, 0, 0, 0, 0
1308     , 0, 0
1309     session1_usage, session2_usage, session3_usage, session4_usage,
1310     session5_usage, session6_usage, session14_usage = 0, 0, 0, 0, 0, 0, 0
1311     session7_usage, session8_usage, session9_usage, session10_usage,
1312     session11_usage, session12_usage, session13_usage = 0, 0, 0, 0, 0, 0, 0
1313     time8 = Time_management.query.filter_by(start_time=8).all()
1314     for timetable in time8:
1315         for booking in timetable.bookings:
1316             session1_income += booking.fees
1317             session1_usage += booking.number
1318
1319     time9 = Time_management.query.filter_by(start_time=9).all()
1320     for timetable in time9:
1321         for booking in timetable.bookings:
1322             session2_income += booking.fees
1323             session2_usage += booking.number
1324
1325     time10 = Time_management.query.filter_by(start_time=10).all()
1326     for timetable in time10:
1327         for booking in timetable.bookings:
1328             session3_income += booking.fees
1329             session3_usage += booking.number
1330
1331     time11 = Time_management.query.filter_by(start_time=11).all()
1332     for timetable in time11:
1333         for booking in timetable.bookings:
1334             session4_income += booking.fees
1335             session4_usage += booking.number
1336
1337     time12 = Time_management.query.filter_by(start_time=12).all()
1338     for timetable in time12:
1339         for booking in timetable.bookings:
1340             session5_income += booking.fees
1341             session5_usage += booking.number
1342
1343     time13 = Time_management.query.filter_by(start_time=13).all()
1344     for timetable in time13:
```

```
1339         for booking in timetable.bookings:
1340             session6_income += booking.fees
1341             session6_usage += booking.number
1342
1343     time14 = Time_management.query.filter_by(start_time=14).all()
1344     for timetable in time14:
1345         for booking in timetable.bookings:
1346             session7_income += booking.fees
1347             session7_usage += booking.number
1348
1349     time15 = Time_management.query.filter_by(start_time=15).all()
1350     for timetable in time15:
1351         for booking in timetable.bookings:
1352             session8_income += booking.fees
1353             session8_usage += booking.number
1354
1355     time16 = Time_management.query.filter_by(start_time=16).all()
1356     for timetable in time16:
1357         for booking in timetable.bookings:
1358             session9_income += booking.fees
1359             session9_usage += booking.number
1360
1361     time17 = Time_management.query.filter_by(start_time=17).all()
1362     for timetable in time17:
1363         for booking in timetable.bookings:
1364             session10_income += booking.fees
1365             session10_usage += booking.number
1366
1367     time18 = Time_management.query.filter_by(start_time=18).all()
1368     for timetable in time18:
1369         for booking in timetable.bookings:
1370             session11_income += booking.fees
1371             session11_usage += booking.number
1372
1373     time19 = Time_management.query.filter_by(start_time=19).all()
1374     for timetable in time19:
1375         for booking in timetable.bookings:
1376             session12_income += booking.fees
1377             session12_usage += booking.number
1378
1379     time20 = Time_management.query.filter_by(start_time=20).all()
1380     for timetable in time20:
1381         for booking in timetable.bookings:
1382             session13_income += booking.fees
1383             session13_usage += booking.number
1384
1385     time21 = Time_management.query.filter_by(start_time=21).all()
1386     for timetable in time21:
1387         for booking in timetable.bookings:
1388             session14_income += booking.fees
```

```
1389         session14_usage += booking.number
1390
1391     overall_income = 0
1392     overall_usage = 0
1393     activities = Activity.query.all()
1394     for activity in activities:
1395         overall_income += activity.weekly_income
1396         overall_usage += activity.weekly_usage
1397     return render_template('business.html', session14_usage=session1
1408 4_usage, session14_income=session14_income,
1398         session13_usage=session13_usage, session1
1409 3_income=session13_income,
1399         session12_usage=session12_usage, session1
1410 2_income=session12_income,
1400         session11_usage=session11_usage, session1
1401         session10_usage=session10_usage, session1
1402         session9_usage=session9_usage, session9_i
1403 ncome=session9_income,
1404         session8_usage=session8_usage, session8_i
1405 ncome=session8_income,
1406         session7_usage=session7_usage, session7_i
1407 ncome=session7_income,
1408         session6_usage=session6_usage,
1409 session6_income=session6_income, session5
1410 _usage=session5_usage,
1411 session5_income=session5_income, session4
1412 _usage=session4_usage,
1413 session4_income=session4_income,
1414 session3_usage=session3_usage, session3_i
1415 ncome=session3_income,
1416 session2_usage=session2_usage, session2_i
1417 ncome=session2_income,
1418 session1_usage=session1_usage, session1_i
1419 ncome=session1_income,
1420         overall_income=overall_income, overall_us
1421 age=overall_usage, activities=activities)
1422
1414 @main.route('/edit-profile', methods=['GET', 'POST'])
1415 @login_required
1416 def edit_profile():
1417     form = EditProfileForm()
1418     if form.validate_on_submit():
1419         current_user.username = form.username.data
1420         db.session.add(current_user._get_current_object())
1421         db.session.commit()
1422         flash('Your profile has been updated.')
```

```
1423         return redirect(url_for('.user', username=current_user.username))
1424     form.name.data = current_user.name
1425     return render_template('edit_profile.html', form=form)
1426
1427 @main.route('/edit-profile/<int:id>', methods=['GET', 'POST'])
1428 @login_required
1429 @admin_required
1430 def edit_profile_admin(id):
1431     user = User.query.get_or_404(id)
1432     form = EditProfileAdminForm(user=user)
1433     if form.validate_on_submit():
1434         user.email = form.email.data
1435         user.username = form.username.data
1436         user.confirmed = form.confirmed.data
1437         user.role = Role.query.get(form.role.data)
1438         db.session.add(user)
1439         db.session.commit()
1440         flash('The profile has been updated.')
1441         return redirect(url_for('.user', username=user.username))
1442     form.email.data = user.email
1443     form.username.data = user.username
1444     form.confirmed.data = user.confirmed
1445     form.role.data = user.role_id
1446     return render_template('edit_profile.html', form=form, user=user)
1447
1448 # app/main/forms.py
1449
1450 from flask_wtf import FlaskForm
1451
1452 from wtforms import StringField, TextAreaField, BooleanField, SelectField, \
1453     SubmitField, IntegerField, RadioField, DateField
1454 from wtforms.validators import DataRequired, Length, Email, Regexp
1455 from ..models import Role, User
1456
1457 class MembershipForm(FlaskForm):
1458     title = SelectField('Title', choices=[('Ms', 'Ms'), ('Mrs', 'Mrs'), ('Mr', 'Mr')], validators=[DataRequired()])
1459     firstname = StringField('First name', validators=[DataRequired(), Length(1, 64), ])
1460     lastname = StringField('Last name', validators=[DataRequired(), Length(1, 64), ])
1461     streetname = StringField('House number and Street name', validators=[DataRequired()])
1462     city = StringField('Town/City', validators=[DataRequired()])
```

```
1465     county = StringField('County', validators=[DataRequired()])
1466     postcode = StringField('Postcode', validators=[DataRequired()])
1467     phone = IntegerField('Area code + Phone number', validators=[DataRequired()])
1468     payment = RadioField('Payment method', choices=[('Credit Card', 'Credit Card'), ('Cash', 'Cash')],
1469                          validators=[DataRequired()])
1470     submit = SubmitField('Submit')
1471
1472 class RefundForm(FlaskForm):
1473     card_number = StringField('Card number', validators=[Length(0, 64)])
1474     expire_month = IntegerField('Expire month', validators=[DataRequired()])
1475     expire_year = IntegerField('Expire year', validators=[DataRequired()])
1476     security_code = IntegerField('Security code', validators=[DataRequired()])
1477     submit = SubmitField('Cancel it now', validators=[DataRequired()])
1478
1479 class BasicBookingForm(FlaskForm):
1480     number = IntegerField('Numbers of people:')
1481     date = DateField('Date(e.g 2020-05-13):', validators=[DataRequired()], format='%Y-%m-%d')
1482     start_time = IntegerField('Start time(e.g 19):', validators=[DataRequired()])
1483     end_time = IntegerField('End time(e.g 20):', validators=[DataRequired()])
1484     payment = RadioField('Payment method', choices=[('Credit Card', 'Credit Card'), ('Cash', 'Cash')],
1485                          validators=[DataRequired()])
1486
1487 class PoolBookingForm(BasicBookingForm):
1488     activity = SelectField('Activity', choices=[('General use', 'General use'), ('Lane swimming', 'Lane swimming'),
1489                                                ('Lessons', 'Lessons'), ('Team events', 'Team events')])
1490     submit = SubmitField('Book')
1491
1492 class FitnessBookingForm(BasicBookingForm):
1493     activity = SelectField('Activity', choices=[('General use', 'General use')])
1494     submit = SubmitField('Book')
1495
1496 class SquashBookingForm(BasicBookingForm):
1497     activity = SelectField('Activity', choices=[('General use', 'General use'), ('Team events', 'Team events')])
```



```
1498     submit = SubmitField('Book')
1499
1500 class HallBookingForm(BasicBookingForm):
1501     activity = SelectField('Activity', choices=[('1-
hour sessions', '1-hour sessions')])
1502     submit = SubmitField('Book')
1503
1504 class EditProfileForm(FlaskForm):
1505     name = StringField('Real name', validators=[Length(0, 64)])
1506     location = StringField('Location', validators=[Length(0, 64)])
1507     about_me = TextAreaField('About me')
1508     submit = SubmitField('Submit')
1509
1510 class CardForm(FlaskForm):
1511     card_number = IntegerField('Card number', validators=[DataRequired()])
1512     holder_name = StringField('Name on account:', validators=[Length(0, 64)])
1513     expire_month = IntegerField('Expire month', validators=[DataRequired()])
1514     expire_year = IntegerField('Expire year', validators=[DataRequired()])
1515     security_code = IntegerField('Security code', validators=[DataRequired()])
1516     submit = SubmitField('Pay securely now', validators=[DataRequired()])
1517
1518 class ConfigureFacilityForm(FlaskForm):
1519     operation = SelectField('Operation', choices=[('add', 'add'), ('edit', 'edit'), ('delete', 'delete')])
1520     name = StringField('Facility name', validators=[Length(0, 64)])
1521     capacity = IntegerField('Capacity', validators=[DataRequired()])
1522     url = StringField('Photo url', validators=[Length(0, 64)])
1523     description = StringField('Description', validators=[Length(0, 64)])
1524     submit = SubmitField('Submit', validators=[DataRequired()])
1525
1526 class ConfigureActivityForm(FlaskForm):
1527     operation = SelectField('Operation', choices=[('add', 'add'), ('edit', 'edit'), ('delete', 'delete')])
1528     facility = StringField('Facility name', validators=[Length(0, 64)])
1529     price = IntegerField('Price', validators=[DataRequired()])
1530     activity = StringField('Activity name', validators=[Length(0, 64)])
1531     submit = SubmitField('Submit', validators=[DataRequired()])
1532
1533 class ConfigureTimetableForm(FlaskForm):
1534     facility = StringField('Facility name', validators=[Length(0, 64)])
```



```
1535     date = DateField('Date',validators=[DataRequired()])
1536     start_time = IntegerField('Open time',validators=[DataRequired()
1537 ])
1537     end_time = IntegerField('Close time',validators=[DataRequired()])
1538     submit = SubmitField('Submit', validators=[DataRequired()])
1539
1540 class SearchForm(FlaskForm):
1541     email = StringField('Email', validators=[DataRequired(), Length(
1542 1, 64),
1543                                     Email()])
1543     submit = SubmitField('Submit', validators=[DataRequired()])
1544
1545 class EditProfileAdminForm(FlaskForm):
1546     email = StringField('Email', validators=[DataRequired(), Length(
1547 1, 64),
1548                                     Email()])
1548     username = StringField('Username', validators=[
1549         DataRequired(), Length(1, 64),
1550         Regexp('^[A-Za-z][A-Za-z0-9_]*$', 0,
1551             'Usernames must have only letters, numbers, dots or '
1552             'underscores')])
1553     confirmed = BooleanField('Confirmed')
1554     role = SelectField('Role', coerce=int)
1555     name = StringField('Real name', validators=[Length(0, 64)])
1556     location = StringField('Location', validators=[Length(0, 64)])
1557     about_me = TextAreaField('About me')
1558     submit = SubmitField('Submit')
1559
1560     def __init__(self, user, *args, **kwargs):
1561         super(EditProfileAdminForm, self).__init__(*args, **kwargs)
1562         self.role.choices = [(role.id, role.name)
1563                               for role in Role.query.order_by(Role.name).all()]
1564         self.user = user
1565
1566     def validate_email(self, field):
1567         if field.data != self.user.email and \
1568             User.query.filter_by(email=field.data).first():
1569             raise ValidationError('Email already registered.')
1570
1571     def validate_username(self, field):
1572         if field.data != self.user.username and \
1573             User.query.filter_by(username=field.data).first():
1574             raise ValidationError('Username already in use.')
1575
1576 /* app/static/style.css */
1577
1578 .profile-thumbnail {
```

```
1579     position: absolute;
1580 }
1581 .profile-header {
1582     min-height: 260px;
1583     margin-left: 280px;
1584 }
1585
1586 <!-- app/templates/403.html -->
1587
1588 {% extends "base.html" %}
1589
1590 {% block title %}Flasky - Forbidden{% endblock %}
1591
1592 {% block page_content %}
1593 <div class="page-header">
1594     <h1>Forbidden</h1>
1595 </div>
1596 {% endblock %}
1597
1598 <!-- app/templates/404.html -->
1599 {% extends "base.html" %}
1600
1601 {% block title %}Flasky - Page Not Found{% endblock %}
1602
1603 {% block page_content %}
1604 <div class="page-header">
1605     <h1>Not Found</h1>
1606 </div>
1607 {% endblock %}
1608
1609 <!-- app/templates/500.html -->
1610 {% extends "base.html" %}
1611
1612 {% block title %}Flasky - Internal Server Error{% endblock %}
1613
1614 {% block page_content %}
1615 <div class="page-header">
1616     <h1>Internal Server Error</h1>
1617 </div>
1618 {% endblock %}
1619
1620 <!-- app/templates/base.html -->
1621 {% extends "bootstrap/base.html" %}
1622
1623 {% block title %}Flasky{% endblock %}
1624
1625 {% block head %}
1626 {{ super() }}
```



```
1662         <li><a href="{ { url_for('main.facility') } }">Facility</a></li>
1663         <li><a href="{ { url_for('main.display_pricing_list') } }">Pricing List</a></li>
1664         <li><a href="{ { url_for('main.search_booking') } }">Search Booking</a></li>
1665         <li><a href="{ { url_for('main.search_booking') } }">Search Membership</a></li>
1666         <li class="dropdown">
1667             <a href="#" class="dropdown-toggle" data-toggle="dropdown">
1668                 Timetable <b class="caret"></b>
1669             </a>
1670             <ul class="dropdown-menu">
1671                 <li><a href="{ { url_for('main.timetable_all', date=0, offset=0) } }">All Timetables</a></li>
1672                 <li><a href="{ { url_for('main.timetable_facility', type=1) } }">One specified facility</a></li>
1673             </ul>
1674         </li>
1675
1676         {% elif current_user.role_id == 3 %}
1677         <li><a href="{ { url_for('main.facility') } }">Facility</a></li>
1678         <li><a href="{ { url_for('main.display_pricing_list') } }">Pricing List</a></li>
1679         <li><a href="{ { url_for('main.view_income') } }">Business</a></li>
1680         <li class="dropdown">
1681             <a href="#" class="dropdown-toggle" data-toggle="dropdown">
1682                 Configure <b class="caret"></b>
1683             </a>
1684             <ul class="dropdown-menu">
1685                 <li><a href="{ { url_for('main.configure_facility') } }">Facility</a></li>
1686                 <li><a href="{ { url_for('main.configure_activity') } }">Activity</a></li>
1687                 <li><a href="{ { url_for('main.configure_timetable') } }">Timetable</a></li>
1688             </ul>
1689         </li>
1690         <li class="dropdown">
1691             <a href="#" class="dropdown-toggle" data-toggle="dropdown">
1692                 Timetable <b class="caret"></b>
1693             </a>
1694             <ul class="dropdown-menu">
1695                 <li><a href="{ { url_for('main.timetable_all', date=0, offset=0) } }">All Timetables</a></li>
```

```

1696             <li><a href="{ { url_for('main.timeta
ble_facility',type=1) } }">One specifed facility</a></li>
1697             </ul>
1698         </li>
1699         {% else %}
1700             <li><a href="{ { url_for('main.facility') } }">F
acity</a></li>
1701             <li><a href="{ { url_for('main.display_pricing_li
st') } }">Pricing List</a></li>
1702             <li><a href="{ { url_for('main.view_membership_ty
pe') } }">Membership</a></li>
1703             <li class="dropdown">
1704                 <a href="#" class="dropdown-toggle" data-
toggle="dropdown">
1705                     Timetable <b class="caret"></b>
1706                 </a>
1707                 <ul class="dropdown-menu">
1708                     <li><a href="{ { url_for('main.timetable_
all', date=0, offset=0) } }">All Timetables</a></li>
1709                     <li><a href="{ { url_for('main.timetable_
facility',type=1) } }">One specifed facility</a></li>
1710                 </ul>
1711             </li>
1712         {% endif %}
1713     </ul>
1714
1715     <ul class="nav navbar-nav navbar-right">
1716         {% if current_user.is_authenticated %}
1717         <li class="dropdown">
1718             <a href="#" class="dropdown-toggle" data-
toggle="dropdown">
1719                 
1720                 Account <b class="caret"></b>
1721             </a>
1722             <ul class="dropdown-menu">
1723                 <li class="page-header">
1724                     <h3>Hello, {% if current_user.is_authenticat
ed %}{ { current_user.username } }{% else %}Stranger{% endif %}!</h3>
1725                 </li>
1726                 {% if current_user.role_id == 1 %}
1727                     <li><a href="{ { url_for('main.display_bookin
gs',id=current_user.id) } }">My bookings</a></li>
1728                     <li><a href="{ { url_for('main.display_cards'
,id=current_user.id) } }">My Card</a></li>
1729                     <li><a href="{ { url_for('main.display_member
ship',id=current_user.id) } }">My Membership</a></li>
1730                 </li>
1731             </li>

```

```
1732         <li><a href="{ url_for('auth.change_password') }}">Change Password</a></li>
1733         <li><a href="{ url_for('auth.change_email_request') }}">Change Email</a></li>
1734         <li><a href="{ url_for('auth.logout') }}">Log Out</a></li>
1735         {% if current_user.is_authenticated %}
1736         <li><a href="{ url_for('main.edit_profile') }}">Change Profile</a></li>
1737         {% endif %}
1738     </ul>
1739 </li>
1740 {% else %}
1741 <li><a href="{ url_for('auth.login') }}">Log In</a>
</li>
1742 {% endif %}
1743 </ul>
1744 </div>
1745 </div>
1746 </div>
1747 {% endblock %}
1748
1749 {% block content %}
1750 <div class="container">
1751     {% for message in get_flashed_messages() %}
1752     <div class="alert alert-warning">
1753         <button type="button" class="close" data-dismiss="alert">&times;</button>
1754         {{ message }}
1755     </div>
1756     {% endfor %}
1757
1758     {% block page_content %}{% endblock %}
1759 </div>
1760 {% endblock %}
1761
1762 {% block scripts %}
1763 {{ super() }}
1764 {{ moment.include_moment() }}
1765 {% endblock %}
1766
1767 <!-- app/templates/book_facility.html -->
1768 {% extends "base.html" %}
1769 {% import "bootstrap/wtf.html" as wtf %}
1770
1771 {% block title %}Book Facility{% endblock %}
1772
1773 {% block page_content %}
1774     <div class="row">
1775         <div class="col-md-6 col-xs-9">
```

```

1776         <h3>Book Facility:</h3>
1777         {{ wtf.quick_form(form) }}
1778     </div>
1779 </div>
1780 {% endblock %}
1781
1782 <!-- app/templates/booking_receipt.html -->
1783 <p>Dear {{ current_user.username }},</p>
1784 <p>Your booking receipt:</p>
1785 <p>*****</p>
1786 <p>Activity: {{ book.activity }}</p>
1787 <p>Date:{{ time.date }}</p>
1788 <p>Time:{{ time.start_time }}:00 ~ {{ time.end_time }}:00</p>
1789 <p>*****</p>
1790 <p>Total: {{book.fees}}</p>
1791 </br>
1792 </br>
1793 </br>
1794 <p>{{ book.timestamp }}</p>
1795 <p></p>
1796
1797 <!-- app/templates/business.html -->
1798 {% extends "base.html" %}
1799 {% import "bootstrap/wtf.html" as wtf %}
1800
1801 {% block title %}Timetable all{% endblock %}
1802
1803 {% block page_content %}
1804
1805 <div class="container-fluid">
1806     <div class="col-md-10 col-md-offset-1 col-xs-10 col-md-
offset-1">
1807
1808         <h3 id="cart">Overall weekly business: </h3>
1809         <h4>Overall income:{{ overall_income }}</h4>
1810         <h4>Overall usage:{{ overall_usage }}</h4>
1811     <br>
1812     <h3 id="cart">Per-activity weekly business: </h3>
1813     <table class="table table-striped table-
hover mytable" id="complete-
table" width='100%' border='0' cellspacing='0' cellpadding='0' style='ta
ble-layout: fixed'>
1814         <tbody>
1815             <tr>
1816                 <th colspan='3' style='word-wrap: break-
word'>Activity</th>
1817                 <th colspan='3' style='word-wrap: break-
word'>Income</th>

```

```

1818                <th colspan='3' style='word-wrap: break-
word'>Usage </th>
1819            </tr>
1820
1821            {% for activity in activities %}
1822                <tr>
1823                    <td colspan='3' style='word-
wrap: break-word'>{{ activity.activity_name}}</td>
1824                    <td colspan='3' style='word-
wrap: break-word'>{{ activity.weekly_income }}</td>
1825                    <td colspan='3' style='word-
wrap: break-word'>{{ activity.weekly_usage }}</td>
1826                </tr>
1827            {% endfor %}
1828        </tbody>
1829    </table>
1830
1831        <h3 id="cart">Per-session weekly business: </h3>
1832        <table class="table table-striped table-
hover mytable" id="complete-
table" width='100%' border='0' cellpadding='0' style='ta
ble-layout: fixed'>
1833            <tbody>
1834                <tr>
1835                    <th colspan='3' style='word-wrap: break-
word'>Session</th>
1836                    <th colspan='3' style='word-wrap: break-
word'>Income</th>
1837                    <th colspan='3' style='word-wrap: break-
word'>Usage </th>
1838                </tr>
1839
1840                <tr>
1841                    <td colspan='3' style='word-wrap: break-
word'>8:00 ~ 9:00</td>
1842                    <td colspan='3' style='word-wrap: break-
word'>{{ session1_income }}</td>
1843                    <td colspan='3' style='word-wrap: break-
word'>{{ session1_usage }}</td>
1844                </tr>
1845
1846                <tr>
1847                    <td colspan='3' style='word-wrap: break-
word'>9:00 ~ 10:00</td>
1848                    <td colspan='3' style='word-wrap: break-
word'>{{ session2_income }}</td>
1849                    <td colspan='3' style='word-wrap: break-
word'>{{ session2_usage }}</td>
1850                </tr>
1851

```



```
1852          <tr>
1853          <td colspan='3' style='word-wrap: break-
word'>10:00 ~ 11:00</td>
1854          <td colspan='3' style='word-wrap: break-
word'>{{ session3_income }}</td>
1855          <td colspan='3' style='word-wrap: break-
word'>{{ session3_usage }}</td>
1856          </tr>
1857
1858          <tr>
1859          <td colspan='3' style='word-wrap: break-
word'>11:00 ~ 12:00</td>
1860          <td colspan='3' style='word-wrap: break-
word'>{{ session4_income }}</td>
1861          <td colspan='3' style='word-wrap: break-
word'>{{ session4_usage }}</td>
1862          </tr>
1863
1864          <tr>
1865          <td colspan='3' style='word-wrap: break-
word'>12:00 ~ 13:00</td>
1866          <td colspan='3' style='word-wrap: break-
word'>{{ session5_income }}</td>
1867          <td colspan='3' style='word-wrap: break-
word'>{{ session5_usage }}</td>
1868          </tr>
1869
1870          <tr>
1871          <td colspan='3' style='word-wrap: break-
word'>13:00 ~ 14:00</td>
1872          <td colspan='3' style='word-wrap: break-
word'>{{ session6_income }}</td>
1873          <td colspan='3' style='word-wrap: break-
word'>{{ session6_usage }}</td>
1874          </tr>
1875          <tr>
1876          <td colspan='3' style='word-wrap: break-
word'>14:00 ~ 15:00</td>
1877          <td colspan='3' style='word-wrap: break-
word'>{{ session7_income }}</td>
1878          <td colspan='3' style='word-wrap: break-
word'>{{ session7_usage }}</td>
1879          </tr>
1880          <tr>
1881          <td colspan='3' style='word-wrap: break-
word'>15:00 ~ 16:00</td>
1882          <td colspan='3' style='word-wrap: break-
word'>{{ session8_income }}</td>
1883          <td colspan='3' style='word-wrap: break-
word'>{{ session8_usage }}</td>
```

```
1884         </tr>
1885         <tr>
1886             <td colspan='3' style='word-wrap: break-
word'>16:00 ~ 17:00</td>
1887             <td colspan='3' style='word-wrap: break-
word'>{{ session9_income }}</td>
1888             <td colspan='3' style='word-wrap: break-
word'>{{ session9_usage }}</td>
1889         </tr>
1890         <tr>
1891             <td colspan='3' style='word-wrap: break-
word'>17:00 ~ 18:00</td>
1892             <td colspan='3' style='word-wrap: break-
word'>{{ session10_income }}</td>
1893             <td colspan='3' style='word-wrap: break-
word'>{{ session10_usage }}</td>
1894         </tr>
1895
1896         <tr>
1897             <td colspan='3' style='word-wrap: break-
word'>18:00 ~ 19:00</td>
1898             <td colspan='3' style='word-wrap: break-
word'>{{ session11_income }}</td>
1899             <td colspan='3' style='word-wrap: break-
word'>{{ session11_usage }}</td>
1900         </tr>
1901
1902         <tr>
1903             <td colspan='3' style='word-wrap: break-
word'>19:00 ~ 20:00</td>
1904             <td colspan='3' style='word-wrap: break-
word'>{{ session12_income }}</td>
1905             <td colspan='3' style='word-wrap: break-
word'>{{ session12_usage }}</td>
1906         </tr>
1907         <tr>
1908             <td colspan='3' style='word-wrap: break-
word'>20:00 ~ 21:00</td>
1909             <td colspan='3' style='word-wrap: break-
word'>{{ session13_income }}</td>
1910             <td colspan='3' style='word-wrap: break-
word'>{{ session13_usage }}</td>
1911         </tr>
1912         <tr>
1913             <td colspan='3' style='word-wrap: break-
word'>21:00 ~ 22:00</td>
1914             <td colspan='3' style='word-wrap: break-
word'>{{ session14_income }}</td>
1915             <td colspan='3' style='word-wrap: break-
word'>{{ session14_usage }}</td>
```

```
1916                                     </tr>
1917
1918                                     </tbody>
1919                                 </table>
1920                             </div>
1921 </div>
1922 {% endblock %}
1923
1924 <!-- app/templates/cancel_booking.html -->
1925 {% extends "base.html" %}
1926 {% import "bootstrap/wtf.html" as wtf %}
1927
1928 {% block title %}Cancel Membership{% endblock %}
1929
1930 {% block page_content %}
1931     <div class="row">
1932         <div class="col-md-6 col-xs-9">
1933             <h3>Booking information:</h3>
1934             <h4>Number: {{ booking.number }}</h4>
1935             <h4>Activity: {{ booking.activity }}</h4>
1936             <h4>Date: {{ time.date }}</h4>
1937             <h4>Time: {{ time.start_time }}:00 ~ {{ time.end_time }}
:00</h4>
1938
1939             <h3>Refund: ${{ booking.fees }}</h3>
1940
1941             <h3>Refund information:</h3>
1942             {{ wtf.quick_form(form) }}
1943         </div>
1944     </div>
1945 {% endblock %}
1946
1947 <!-- app/templates/cancel_membership.html -->
1948 {% extends "base.html" %}
1949 {% import "bootstrap/wtf.html" as wtf %}
1950
1951 {% block title %}Cancel Membership{% endblock %}
1952
1953 {% block page_content %}
1954     <div class="row">
1955         <div class="col-md-6 col-xs-9">
1956             <h3>Membership information:</h3>
1957             <h4>Status: {{ membership.status }}</h4>
```

```
1958         <h4>Start date: {{ membership.start_date }}</h4>
1959         <h4>End date: {{ membership.end_date }}</h4>
1960         <h4>Days left: {{ days_left }}</h4>
1961
1962         <h3>Refund information:</h3>
1963         {{ wtf.quick_form(form) }}
1964     </div>
1965 </div>
1966     <div class="col-md-4 col-xs-6">
1967         <h3>Refund: ${{ money_refund }}</h3>
1968     </div>
1969 </div>
1970
1971 {% endblock %}
1972
1973 <!-- app/templates/configure_activity.html -->
1974 {% extends "base.html" %}
1975 {% import "bootstrap/wtf.html" as wtf %}
1976
1977 {% block title %}Configure Activity{% endblock %}
1978
1979 {% block page_content %}
1980     <div class="row">
1981         <div class="col-md-6 col-xs-9">
1982             <h3>Configure Activity:</h3>
1983             {{ wtf.quick_form(form) }}
1984         </div>
1985     </div>
1986 {% endblock %}
1987
1988 <!-- app/templates/configure_facility.html -->
1989 {% extends "base.html" %}
1990 {% import "bootstrap/wtf.html" as wtf %}
1991
1992 {% block title %}Configure Facility{% endblock %}
1993
1994 {% block page_content %}
1995     <div class="row">
1996         <div class="col-md-6 col-xs-9">
1997             <h3>Configure Facility:</h3>
1998
1999             {{ wtf.quick_form(form) }}
2000
2001         </div>
2002     </div>
2003 {% endblock %}
2004
2005 <!-- app/templates/configure_timetable.html -->
```

```
2006 {% extends "base.html" %}
2007 {% import "bootstrap/wtf.html" as wtf %}
2008
2009 {% block title %}Configure Timetable{% endblock %}
2010
2011 {% block page_content %}
2012     <div class="row">
2013         <div class="col-md-6 col-xs-9">
2014             <h3>Configure Timetable:</h3>
2015
2016                 {{ wtf.quick_form(form) }}
2017
2018         </div>
2019     </div>
2020 {% endblock %}
2021
2022 <!-- app/templates/edit_profile.html -->
2023 {% extends "base.html" %}
2024 {% import "bootstrap/wtf.html" as wtf %}
2025
2026 {% block title %}Flasky - Edit Profile{% endblock %}
2027
2028 {% block page_content %}
2029 <div class="page-header">
2030     <h1>Edit Your Profile</h1>
2031 </div>
2032 <div class="col-md-4">
2033     {{ wtf.quick_form(form) }}
2034 </div>
2035 {% endblock %}
2036
2037 <!-- app/templates/facility.html -->
2038 {% extends "base.html" %}
2039
2040 {% block title %}Facility{% endblock %}
2041
2042 {% block page_content %}
2043     <div class="container-fluid">
2044         {% for facility in facilities %}
2045             <div class="col-md-4 col-xs-6 ">
2046                 <a href="{{ url_for('.view_info', id=facility.id) }}"
2047 >
2048                     
2049                 </a>
2050                 <a href="{{ url_for('.view_info',id=facility.id) }}">
2051                     <h4>{{ facility.name }}</h4>
2052                 </a>
```

```
2052         <a href="{ { url_for('.view_info',id=facility.id) }}" >
2053             <h5>View more</h5>
2054         </a>
2055     </div>
2056     {% endfor %}
2057 </div>
2058 {% endblock %}
2059
2060 <!-- app/templates/facility_info.html -->
2061 {% extends "base.html" %}
2062
2063 {% block title %}Facility Detail{% endblock %}
2064
2065 {% block page_content %}
2066 <div class="good-form">
2067     <div class="row">
2068         <div class="col-md-5 col-md-offset-1 col-xs-7 col-md-offset-
2069 2">
2069             
2070         </div>
2071         <div class="col-md-5 col-xs-6">
2072             <h3>Name:</br>{ { facility.name } }</h3>
2073             <h3>Description:</br>{ { facility.description } }</h3>
2074             <h3>Capacity:</br>{ { facility.capacity } }</h3>
2075             <h3>Activity:
2076             {% for activity in facility.activities %}
2077                 <br>
2078                 { { activity.activity_name } }
2079             {% endfor %}
2080             </h3>
2081         </div>
2082     </div>
2083 </div>
2084
2085 {% endblock %}
2086
2087 <!-- app/templates/handle_card_booking.html -->
2088 {% extends "base.html" %}
2089 {% import "bootstrap/wtf.html" as wtf %}
2090
2091 {% block title %}Membership{% endblock %}
2092
2093 {% block page_content %}
2094     <div class="row">
2095         <div class="col-md-6 col-xs-9">
2096             <h3>Booking information:</h3>
2097             <h4>Number: { { book.number } }</h4>
2098             <h4>Activity: { { book.activity } }</h4>
2099         </div>
2100     </div>
2101 </div>
2102 {% endblock %}
```

```

2099         <h4>Date: {{ time.date }}</h4>
2100         <h4>Time: {{time.start_time }}:00 ~ {{ time.end_time }}:
00</h4>
2101         <h4>Total: ${{ book.fees }}</h4>
2102
2103         <h3>Payment information:</h3>
2104         {{ wtf.quick_form(form) }}
2105
2106     </div>
2107
2108 {% endblock %}
2109
2110 <!-- app/templates/index.html -->
2111 {% extends "base.html" %}
2112
2113 {% block head %}
2114 {{ super() }}
2115 <link rel="stylesheet" type="text/css" href="https://cdn.jsdelivrivr.n
et/npm/cookieconsent@3/build/cookieconsent.min.css" />
2116 {% endblock %}
2117
2118 {% block scripts %}
2119 {{ super() }}
2120
<script src="https://cdn.jsdelivrivr.net/npm/cookieconsent@3/build/cookiecon
sent.min.js" data-cfasync="false"></script>
2121 <script>
2122     window.cookieconsent.initialise({
2123         "palette": {
2124             "popup": {
2125                 "background": "#000"
2126             },
2127             "button": {
2128                 "background": "#f1d600"
2129             }
2130         },
2131         "theme": "classic"
2132     });
2133 </script>
2134 {% endblock %}
2135
2136 {% block page_content %}
2137     <div class="container"><br>
2138         <div class="col-md-8 col-md-offset-1 col-xs-10 col-xs-
offset-1">
2139             <div id="myCarousel" class="carousel slide center-
block" style="width:900px;">
2140                 <!-- 轮播 (Carousel) 指标 -->
2141                 <ol class="carousel-indicators">

```

```
2142         <li data-target="#myCarousel" data-slide-
to="0" class="active"></li>
2143         <li data-target="#myCarousel" data-slide-
to="1"></li>
2144         <li data-target="#myCarousel" data-slide-
to="2"></li>
2145     </ol>
2146
2147     <!-- (Carousel) Project -->
2148     <div class="carousel-inner">
2149         <div class="item active">
2150             
2151             <div class="carousel-caption">
2152                 <!--<h3>Movie Information</h3>
2153                 <p>bravo!</p>-->
2154             </div>
2155         </div>
2156         <div class="item">
2157             
2158         </div>
2159         <div class="item">
2160             
2161         </div>
2162         <div class="item">
2163             
2164         </div>
2165     </div>
2166
2167     <a href="#myCarousel" class="left carousel-
control" data-slide="prev">
2168         <span class="glyphicon glyphicon-chevron-
left"></span>
2169     </a>
2170     <a href="#myCarousel" class="right carousel-
control" data-slide="next">
2171         <span class="glyphicon glyphicon-chevron-
right"></span>
2172     </a>
2173 </div>
2174 </div>
2175 </div>
2176
2177 <script>
2178     $('#myCarousel').carousel({
2179         interval:2000,
2180         wrap:true
```



```
2181     }).on('slide.bs.carousel', function () {
2182         //alert('slide')
2183     }).on('slid.bs.carousel', function () {
2184         //alert('slid')
2185     })
2186     $('#btn').click(function(){
2187         //$('#myCarousel').carousel(2);
2188     });
2189 </script>
2190
2191 <div class="center-block">
2192 <div class=" col-md-7 col-md-offset-2 col-xs-10 col-xs-offset-
2193 1">
2194     <h1>Hello, {% if current_user.is_authenticated %}{{ current_
2195 user.username }}{% else %}Stranger{% endif %}!
2196     Welcome to Gym.</h1>
2197
2198     <h3>Swimming Pool</h3>
2199 <p>Our main swimming pool is great for fun and fitness alike. With both s
2200 hallow and deep ends, this pool is widely accessible for all ages and cap
2201 abilities. We have a wide variety of swimming lessons for both adults and
2202 children as well as a timetable of varied activities.
2203 </p>
2204 <h3>Premier Suite (Sauna / Steam room)</h3>
2205 <p>Our premier suite is the ideal place to unwind and relax after a long
2206 day or intense workout. This is an adult only zone, with its own changing
2207 area located on the pool level.
2208 </p>
2209 <h3>Gym</h3>
2210 <p>Our gym was fully refurbished and re-
2211 opened in October 2012. Because of this the gym now offers a bigger and b
2212 etter fitness experience. The upgrade has seen the gym double in size and
2213 has made room for brand new state of the art gym equipment. New flooring
2214 , air conditioning and paintwork have been added to compliment the upgrad
2215 e. This gym offers 40 stations including but not limited to:
2216
2217 Treadmills
2218 X trainers
2219 Bikes
2220 Rowing machines
2221 Steppers
2222 Cable machines
2223
2224 Selected Technogym equipment have audio jacks that allow you to connect t
2225 o the in gym entertainment system. Please note: an induction is required
2226 for your own safety before using the gym for the first time.
2227 </p>
```

2213 <h3>Sports hall</h3>

2214

<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.

2215 </p>

2216 <h3>Outdoor Pitch</h3>

2217

<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.

2218 </p>

2219 <h3>Childrens Parties</h3>

2220

<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.

2221 </p>

2222 <h3>Parking</h3>

2223

<p>We have 2 car parks and 6 disabled spots accommodating over 100 vehicles. We also have a undercover cycle rack.</p>

2224 </div>

2225 </div>

2226

2227 {% endblock %}

2228

2229

2230 <!-- app/templates/my_booking.html -->

2231 {% extends "base.html" %}

2232

2233 {% block title %}My booking{% endblock %}

2234

2235 {% block page_content %}

2236 {% if number == 0 %}

2237 <div class="col-md-offset-4">

2238 You have no bookings.

2239 </div>

2240 {% else %}

2241 <h4 id="cart">You have {{ number }} booking histories!</h4>

2242 {% endif %}

2243

2244 <div class="container-fluid">

2245 {% for booking in bookings %}

2246 <div class="col-md-10 col-md-offset-1 col-xs-10 col-md-offset-1">

```

2247         <a href="../../static/{{booking.id}}.pdf" download="receipt.pdf">Download the receipt</a>
2248     </div>
2249     <div class="col-md-10 col-md-offset-1 col-xs-10 col-md-
offset-1">
2250         <table class="table table-striped table-
hover mytable" id="complete-
table" width='100%' border='0' cellspacing='0' cellpadding='0' style='ta
ble-layout: fixed'>
2251             <tbody>
2252                 <tr>
2253                     <th colspan='3' style='word-wrap: break-
word'>Booking id:{{booking.id}}</th>
2254                     <th colspan='3' style='word-wrap: break-
word'>Price:${{booking.fees}}.0</th>
2255                     <th colspan='3' style='word-wrap: break-
word'>Status:{{booking.status }}</th>
2256                     <th colspan='3' style='word-wrap: break-
word'>Payment:{{booking.payment}}</th>
2257                     <th colspan='3' style='word-wrap: break-
word'>{{booking.timestamp }}</th>
2258                     {% if booking.status == "Unpaid" %}
2259                     <th colspan='3' style='word-
wrap: break-word'>
2260                         <a class="btn btn-mini btn-
success" href="{{url_for('.handle_card_booking',book_id=booking.id)}}">Pa
y now</a>
2261                     </th>
2262                     <th colspan='3' style='word-
wrap: break-word'>
2263                         <a class="btn btn-mini btn-
info" href="{{url_for('.cancel_booking',id=booking.id)}}">Cancel membersh
ip</a>
2264                     </th>
2265                     {% elif booking.status == "Paid" %}
2266                     <th colspan='3' style='word-
wrap: break-word'>
2267                         <a class="btn btn-mini btn-
info" href="{{url_for('.cancel_booking',id=booking.id)}}">Cancel membersh
ip</a>
2268                     </th>
2269                     {% endif %}
2270                 </tr>
2271             </tbody>
2272         </table>
2273     </div>
2274     {% for timetable in timetable_booking %}
2275     {% if booking.time_id == timetable.id %}
2276         <div class="row">

```

```
2277         <div class="col-md-4 col-md-offset-
2278 2 col-xs-6 col-xs-offset-4">
2279         <h3>Booking content:</h3>
2280         {% for facility in facilities %}
2281         {% if facility.id == timetab
le.facility_id %}
2282         <span>Facility:{{ facil
ity.name }} <br> Date: {{timetable.date }}<br>Time:{{timetable.start_time
}}:00 ~ {{timetable.end_time }}:00<br></span>
2283         {% endif %}
2284         {% endfor %}
2285     </div>
2286 </div>
2287 {% endif %}
2288 {% endfor %}
2289
2290 {% endfor %}
2291 </div>
2292
2293 {% endblock %}
2294
2295 # migrations/env.py
2296 from __future__ import with_statement
2297 from alembic import context
2298 from sqlalchemy import engine_from_config, pool
2299 from logging.config import fileConfig
2300
2301 config = context.config
2302
2303 fileConfig(config.config_file_name)
2304
2305 from flask import current_app
2306
2307 config.set_main_option('sqlalchemy.url', current_app.config.get('SQLALCHE
MY_DATABASE_URI'))
2308 target_metadata = current_app.extensions['migrate'].db.metadata
2309
2310 def run_migrations_offline():
2311     url = config.get_main_option("sqlalchemy.url")
2312     context.configure(url=url)
2313
2314     with context.begin_transaction():
2315         context.run_migrations()
2316
2317 def run_migrations_online():
2318     engine = engine_from_config(
2319         config.get_section(config.config_ini_section),
2320         prefix='sqlalchemy.',
```

```
2320             poolclass=pool.NullPool)
2321
2322     connection = engine.connect()
2323     context.configure(
2324         connection=connection,
2325         target_metadata=target_metadata
2326     )
2327
2328     try:
2329         with context.begin_transaction():
2330             context.run_migrations()
2331     finally:
2332         connection.close()
2333
2334 if context.is_offline_mode():
2335     run_migrations_offline()
2336 else:
2337     run_migrations_online()
2338
2339 # migrations/alembic.ini
2340
2341 [alembic]
2342 # template used to generate migration files
2343 # file_template = %(rev)s_%(slug)s
2344
2345 # set to 'true' to run the environment during
2346 # the 'revision' command, regardless of autogenerate
2347 # revision_environment = false
2348
2349 # Logging configuration
2350 [loggers]
2351 keys = root,sqlalchemy,alembic
2352
2353 [handlers]
2354 keys = console
2355
2356 [formatters]
2357 keys = generic
2358
2359 [logger_root]
2360 level = WARN
2361 handlers = console
2362 qualname =
2363
2364 [logger_sqlalchemy]
2365 level = WARN
2366 handlers =
2367 qualname = sqlalchemy.engine
2368
```

```
2369 [logger_alembic]
2370 level = INFO
2371 handlers =
2372 qualname = alembic
2373
2374 [handler_console]
2375 class = StreamHandler
2376 args = (sys.stderr,)
2377 level = NOTSET
2378 formatter = generic
2379
2380 [formatter_generic]
2381 format = %(levelname)-5.5s [% (name)s] %(message)s
2382 datefmt = %H:%M:%S
2383
2384 # migrations/script.py.mako
2385
2386 """${message}
2387
2388 Revision ID: ${up_revision}
2389 Revises: ${down_revision}
2390 Create Date: ${create_date}
2391
2392 """
2393
2394 # revision identifiers, used by Alembic.
2395 revision = ${repr(up_revision)}
2396 down_revision = ${repr(down_revision)}
2397
2398 from alembic import op
2399 import sqlalchemy as sa
2400 ${imports if imports else ""}
2401
2402 def upgrade():
2403     ${upgrades if upgrades else "pass"}
2404
2405
2406 def downgrade():
2407     ${downgrades if downgrades else "pass"}
2408
2409 # test/test_basics.py
2410 import unittest
2411 from flask import current_app
2412 from app import create_app, db
2413
2414 class BasicsTestCase(unittest.TestCase):
2415     def setUp(self):
2416         self.app = create_app('testing')
2417         self.app_context = self.app.app_context()
```

```
2418         self.app_context.push()
2419         db.create_all()
2420
2421     def tearDown(self):
2422         db.session.remove()
2423         db.drop_all()
2424         self.app_context.pop()
2425
2426     def test_app_exists(self):
2427         self.assertFalse(current_app is None)
2428
2429     def test_app_is_testing(self):
2430         self.assertTrue(current_app.config['TESTING'])
2431
2432 # test/test_user_model.py
2433 import unittest
2434 import time
2435 from datetime import datetime
2436 from app import create_app, db
2437 from app.models import User, AnonymousUser, Role, Permission
2438
2439 class UserModelTestCase(unittest.TestCase):
2440     def setUp(self):
2441         self.app = create_app('testing')
2442         self.app_context = self.app.app_context()
2443         self.app_context.push()
2444         db.create_all()
2445         Role.insert_roles()
2446
2447     def tearDown(self):
2448         db.session.remove()
2449         db.drop_all()
2450         self.app_context.pop()
2451
2452     def test_password_setter(self):
2453         u = User(password='cat')
2454         self.assertTrue(u.password_hash is not None)
2455
2456     def test_no_password_getter(self):
2457         u = User(password='cat')
2458         with self.assertRaises(AttributeError):
2459             u.password
2460
2461     def test_password_verification(self):
2462         u = User(password='cat')
2463         self.assertTrue(u.verify_password('cat'))
2464         self.assertFalse(u.verify_password('dog'))
2465
2466     def test_password_salts_are_random(self):
```

```
2467         u = User(password='cat')
2468         u2 = User(password='cat')
2469         self.assertTrue(u.password_hash != u2.password_hash)
2470
2471     def test_valid_confirmation_token(self):
2472         u = User(password='cat')
2473         db.session.add(u)
2474         db.session.commit()
2475         token = u.generate_confirmation_token()
2476         self.assertTrue(u.confirm(token))
2477
2478     def test_invalid_confirmation_token(self):
2479         u1 = User(password='cat')
2480         u2 = User(password='dog')
2481         db.session.add(u1)
2482         db.session.add(u2)
2483         db.session.commit()
2484         token = u1.generate_confirmation_token()
2485         self.assertFalse(u2.confirm(token))
2486
2487     def test_expired_confirmation_token(self):
2488         u = User(password='cat')
2489         db.session.add(u)
2490         db.session.commit()
2491         token = u.generate_confirmation_token(1)
2492         time.sleep(2)
2493         self.assertFalse(u.confirm(token))
2494
2495     def test_valid_reset_token(self):
2496         u = User(password='cat')
2497         db.session.add(u)
2498         db.session.commit()
2499         token = u.generate_reset_token()
2500         self.assertTrue(User.reset_password(token, 'dog'))
2501         self.assertTrue(u.verify_password('dog'))
2502
2503     def test_invalid_reset_token(self):
2504         u = User(password='cat')
2505         db.session.add(u)
2506         db.session.commit()
2507         token = u.generate_reset_token()
2508         self.assertFalse(User.reset_password(token + 'a', 'horse'))
2509         self.assertTrue(u.verify_password('cat'))
2510
2511     def test_valid_email_change_token(self):
2512         u = User(email='john@example.com', password='cat')
2513         db.session.add(u)
2514         db.session.commit()
2515         token = u.generate_email_change_token('susan@example.org')
2516         self.assertTrue(u.change_email(token))
```


目录

一、	引言.....	2
二、	软件概述.....	2
1.	软件结构.....	2
2.	功能.....	4
三、	运行环境.....	5
1.	硬件环境.....	5
2.	软件环境.....	5
四、	技术细节.....	6
1.	解决方案摘要.....	6
2.	数据建模.....	6
3.	数据库设计.....	9
4.	部署方案.....	10
5.	用户使用方案.....	11
6.	实现机制.....	14
五、	用户指南.....	15
1.	安装软件并启动.....	15
2.	注册账号.....	15
3.	管理员.....	18
4.	普通用户.....	23

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	foreignkey 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			

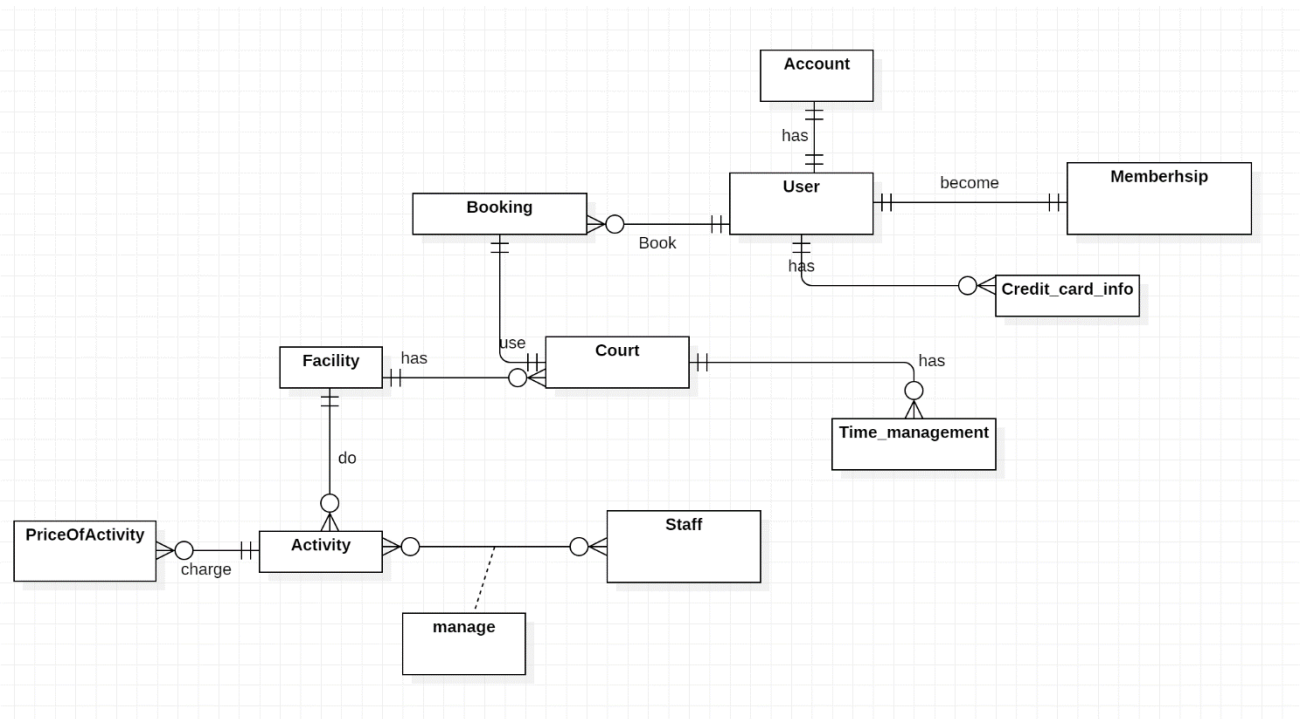
Credit_card_info Model

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

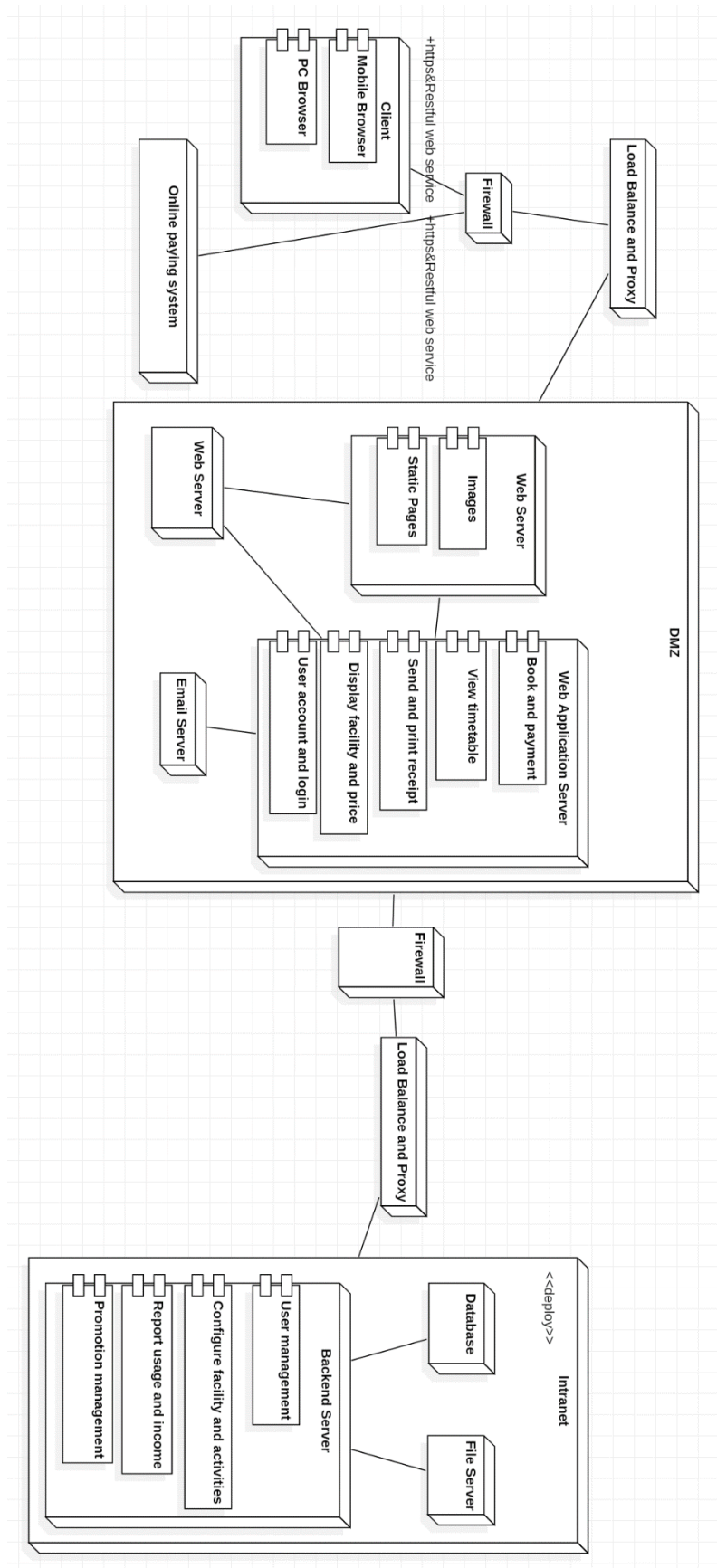
Account Model

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



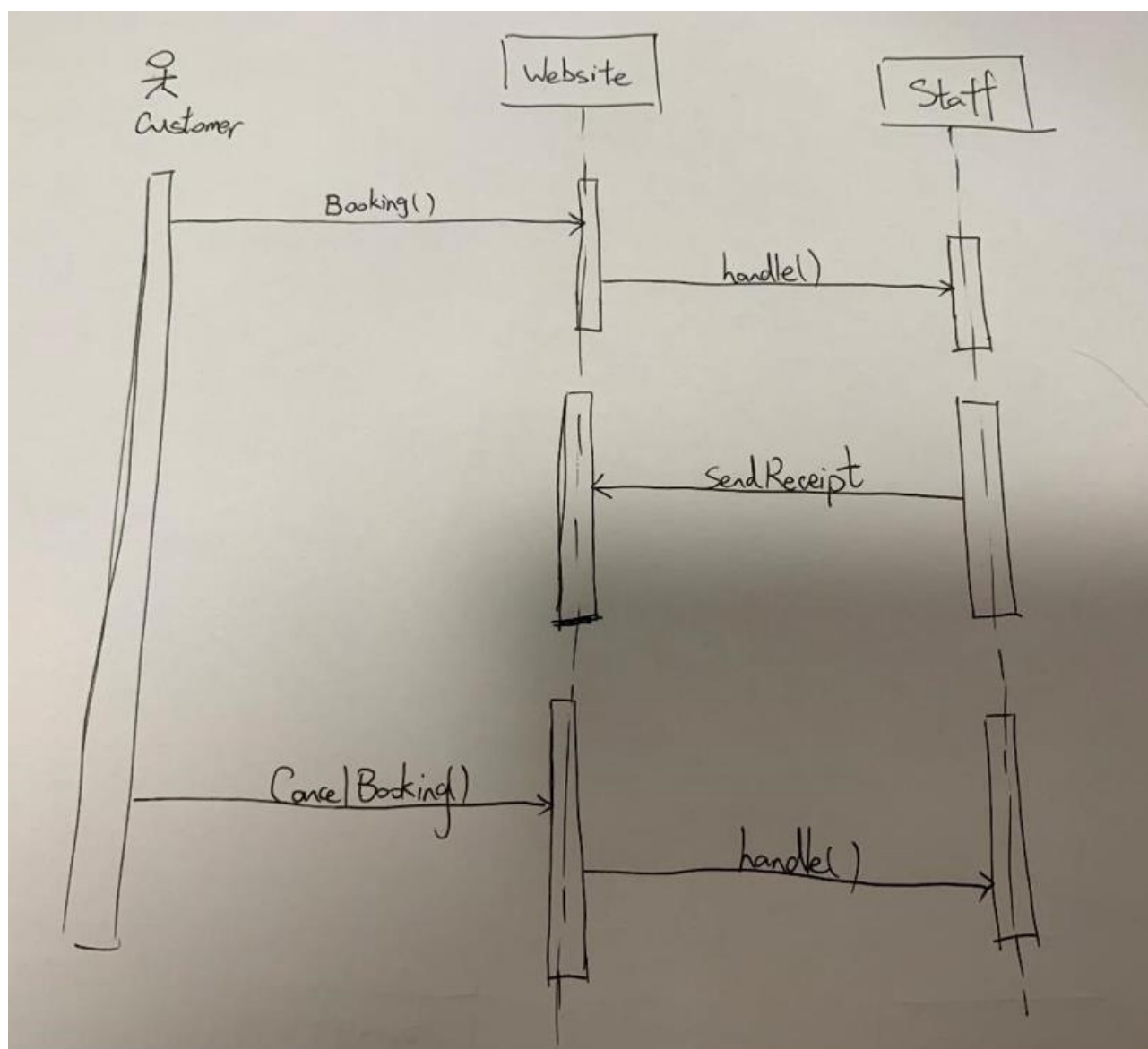
4. 部署方案



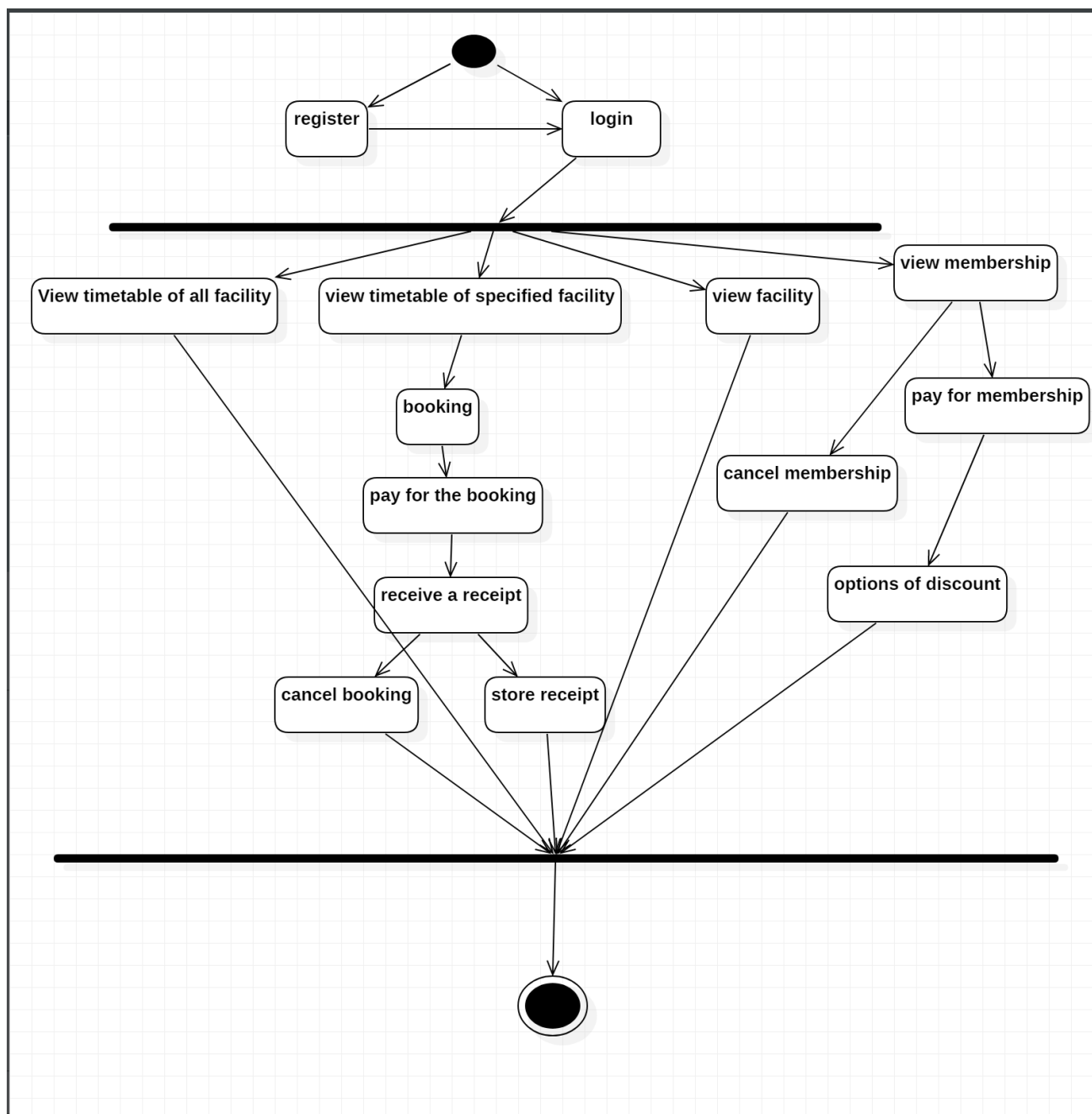
硬件包括智能手机、各种服务器、防火墙和工作负载平衡。该软件包括客户端应用程序/浏览器、DMZ 中的 web 服务器、web 应用服务器、数据库、应用管理后端服务器和电子邮件服务器。

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。

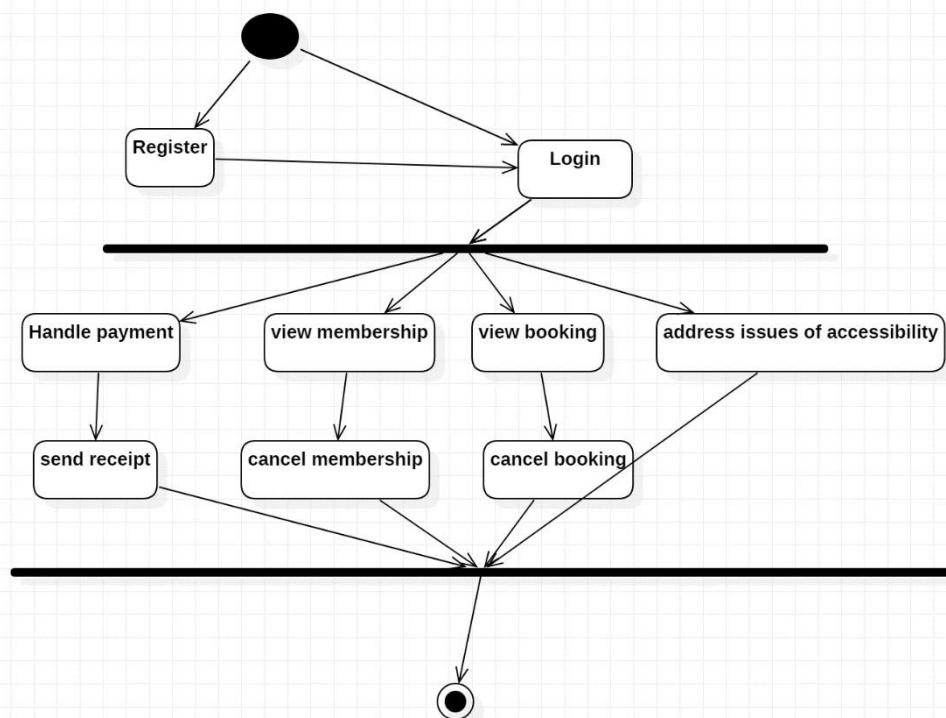
5. 用户使用方案



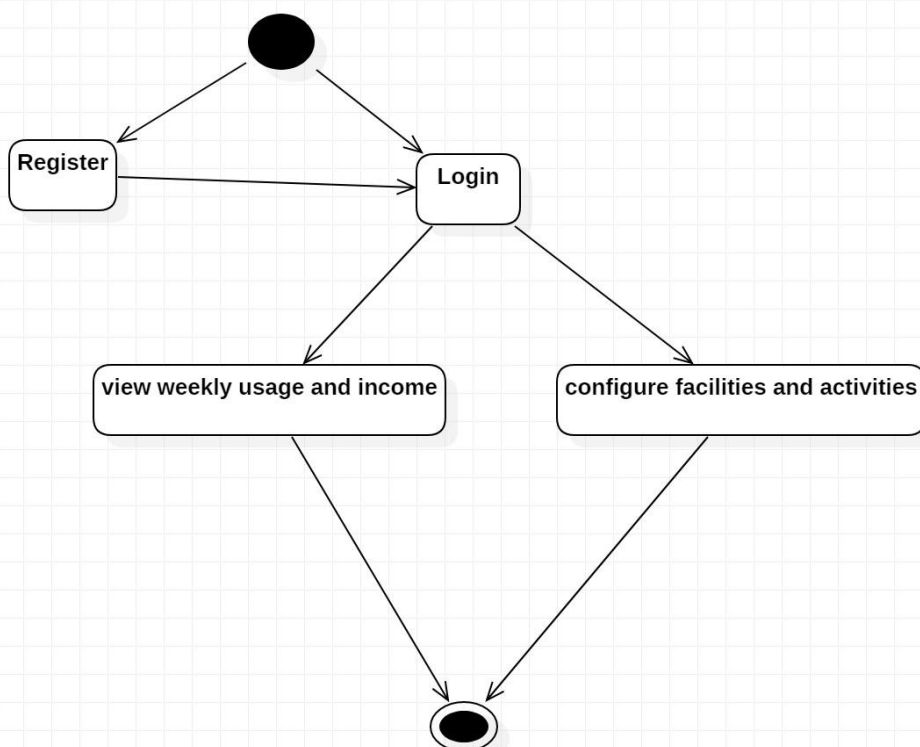
我是小明，我是体育中心的会员。我想打篮球，所以我访问了体育中心的网站。我想看一下篮球场的时间表，以便知道我什么时候可以去打篮球。我想订一个篮球场，以便有一天我能打篮球。



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。



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。SQLAlchemy 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。

Facade-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 facade 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
```

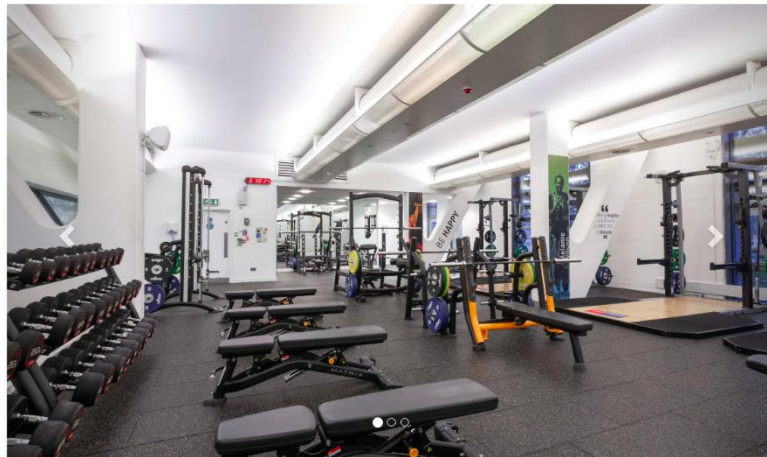
然后打开 flask shell，输入 models.Role.insert_roles()来将角色 ID 添加至数据库中。

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. [Learn more](#)

Got it!



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：

The GymFacilityPricing ListMembershipTimetable

Log In

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：

The GymFacilityPricing ListMembershipTimetable

Log In

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：

The GymFacilityPricing ListMembershipTimetable

Account

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/eyJhbGciOiJIUzI1NiIsImIhdCI6MTU5MDQyMTkwMCwiZXhwIjo_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：

The Gym Facilities Pricing List Membership Timetable ▾

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

Register

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

Submit

点击 Activities，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

[+ Add activity](#)

Edit Activity

Staff ID number

Price of Activity

Activity Name

Facility ID

点击 Membership 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

[+ Add membership type](#)

Edit Membership Type

Name

Length in days

Price

点击 Membership，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


[+ Add activity instance](#)

Edit Activity Instance

Start time

End time

Activity ID

Court ID

Submit

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

Join us now

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)

VIP

£45.00

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

Submit

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.



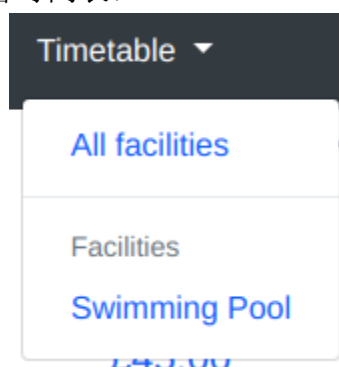
Thanks! You have successfully purchased a VIP membership.



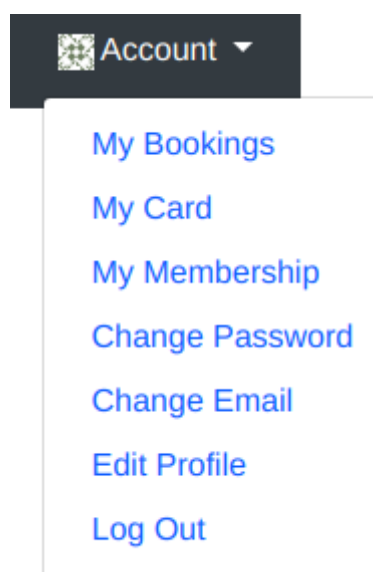
Receipt: Payer:aria item:General use Fees:5 2020-05-13 16:56:02



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	<button>Cancel membership</button>

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:

① 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)

Submit

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

Update Email Address

Change Your Password

Old password

New password

Confirm new password

Update Password