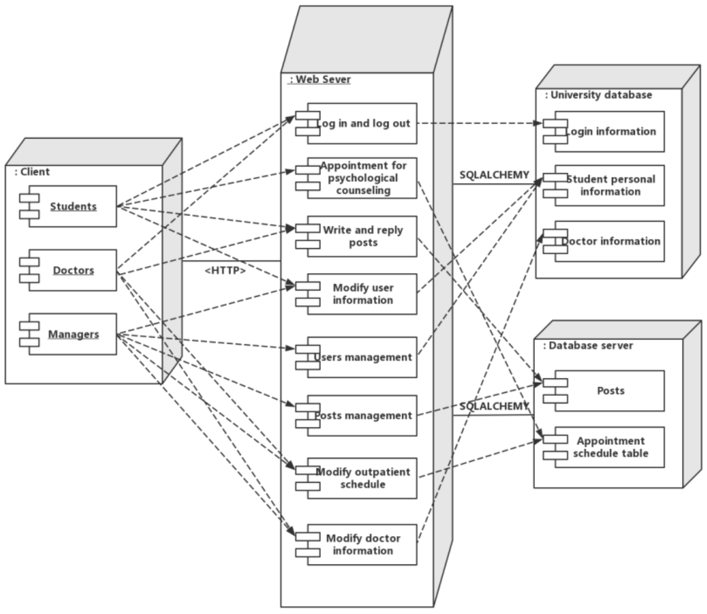
1 Overview of Technical Design



The system uses the Flask framework as the main technical framework, and the SQLite database as the data storage unit of the system.

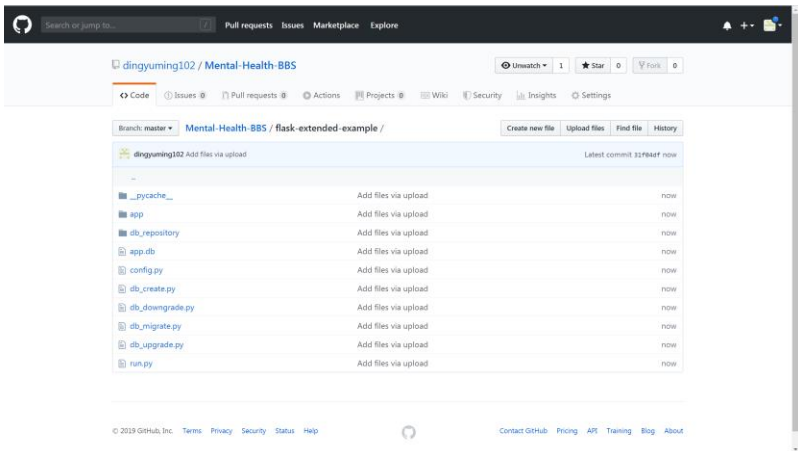
The system adopts the B / S three-layer structure mode as the framework for design and implementation.

Programming language	Python 3.7
Libraries	flask, flask_sqlalchemy, flask_admin, flask_wtf, wtforms, urllib.parse, time, os, migrate.versioning
Framework	flask 1.1.1
Databases	SQLite3
Editor and IDE	JetBrains PyCharm (Community Edition 2019.2.4 x64)

2 Evidence of development

2.1 Version control

Upload the project to github for version control.



dingyuming102 / Mental-Health-BBS

Branch: master ▾ Mental-Health-BBS / flask-extended-example /

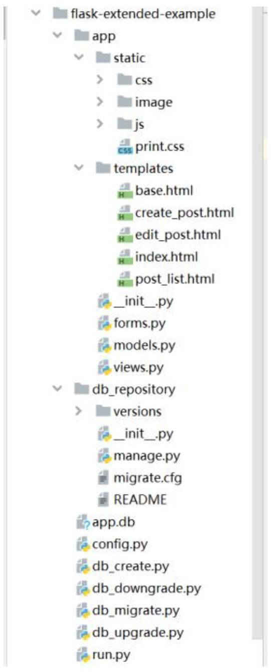
dingyuming102 Add files via upload — Latest commit 31f04df now

__pycache__	Add files via upload	now
app	Add files via upload	now
db_repository	Add files via upload	now
app.db	Add files via upload	now
config.py	Add files via upload	now
db_create.py	Add files via upload	now
db_downgrade.py	Add files via upload	now
db_migrate.py	Add files via upload	now
db_upgrade.py	Add files via upload	now
run.py	Add files via upload	now

© 2019 GitHub, Inc.

2.2 A minimum viable proof of architecture

This project is designed using a three-tier architecture of B / S structure.



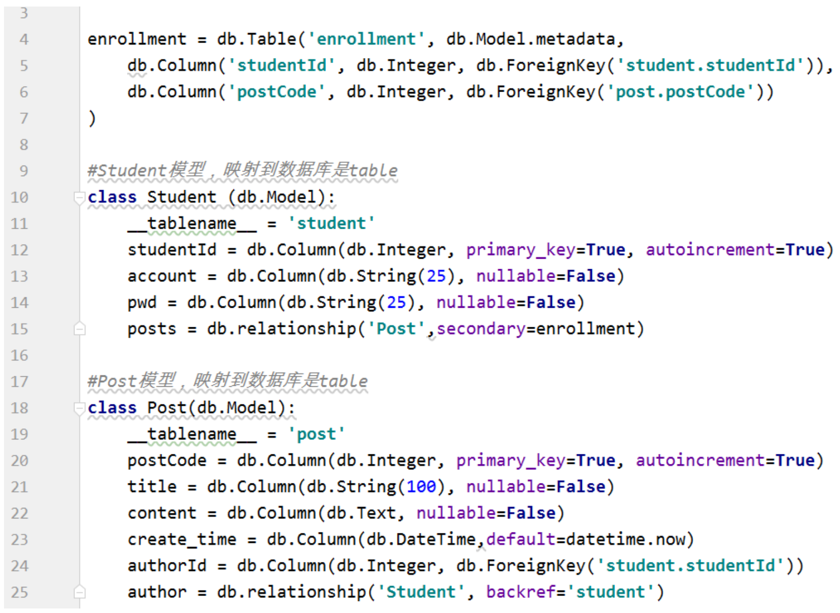
```
flask-extended-example
  app
    static
      css
      image
      js
      print.css
    templates
      base.html
      create_post.html
      edit_post.html
      index.html
      post_list.html
    __init__.py
    forms.py
    models.py
    views.py
  db_repository
    versions
    __init__.py
    manage.py
    migrate.cfg
    README
  app.db
  config.py
  db_create.py
  db_downgrade.py
  db_migrate.py
  db_upgrade.py
  run.py
```

All the html files and static folders under the templates folder belong to the UI layer.

The view functions in the views.py file belong to the business logic layer.

All files in the db_repository file in this project belong to the DAL. The code in these files and files builds the database model in the flask framework.

2.3 RDBMS

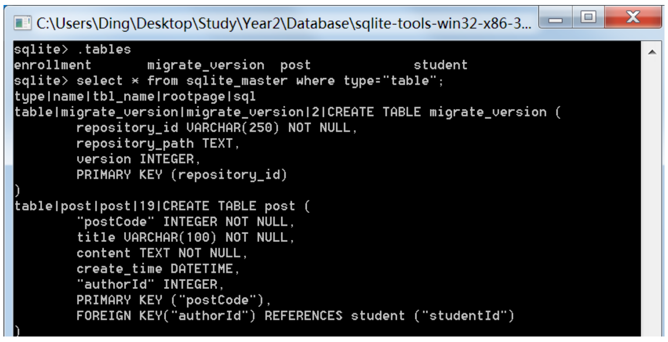


```python

enrollment = db.Table('enrollment', db.Model.metadata,
    db.Column('studentId', db.Integer, db.ForeignKey('student.studentId')),
    db.Column('postCode', db.Integer, db.ForeignKey('post.postCode'))
)

#Student模型，映射到数据库是table
class Student (db.Model):
    __tablename__ = 'student'
    studentId = db.Column(db.Integer, primary_key=True, autoincrement=True)
    account = db.Column(db.String(25), nullable=False)
    pwd = db.Column(db.String(25), nullable=False)
    posts = db.relationship('Post',secondary=enrollment)

#Post模型，映射到数据库是table
class Post(db.Model):
    __tablename__ = 'post'
    postCode = db.Column(db.Integer, primary_key=True, autoincrement=True)
    title = db.Column(db.String(100), nullable=False)
    content = db.Column(db.Text, nullable=False)
    create_time = db.Column(db.DateTime,default=datetime.now)
    authorId = db.Column(db.Integer, db.ForeignKey('student.studentId'))
    author = db.relationship('Student', backref='student')
```

Relational models for 'Student' and 'Post'

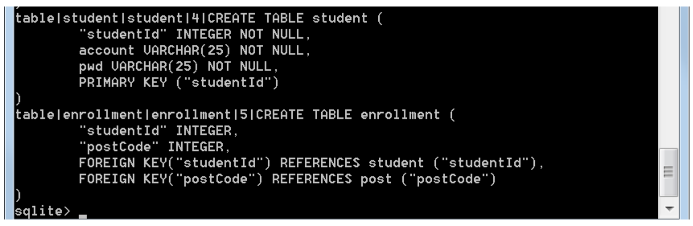
2.4 Unit testing

2.4.1 Test if the relational model is successfully mapped to the database. Whether the table was successfully created.



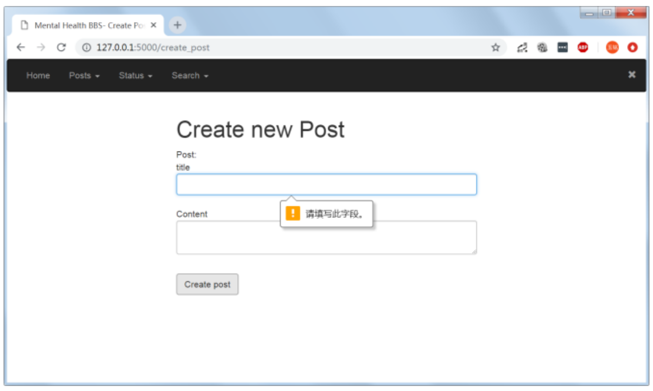
C:\Users\Ding\Desktop\Study\Year2\Database\sqlite-tools-win32-x86-3...

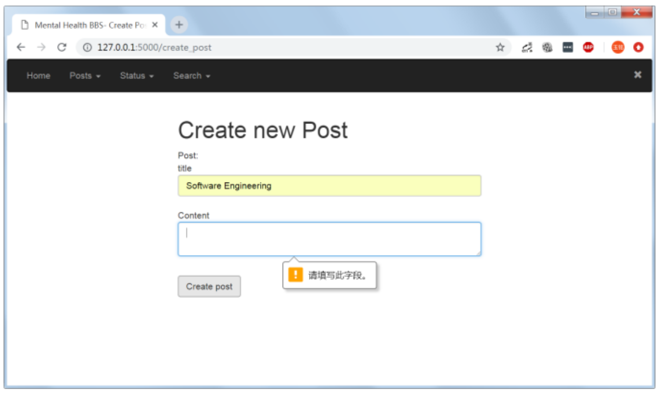
```
sqlite> .tables
enrollment       migrate_version  post             student
sqlite> select * from sqlite_master where type="table";
type|name|tbl_name|rootpage|sql
table|migrate_version|migrate_version|2|CREATE TABLE migrate_version (
        repository_id VARCHAR(250) NOT NULL,
        repository_path TEXT,
        version INTEGER,
        PRIMARY KEY (repository_id)
)
table|post|post|19|CREATE TABLE post (
        "postCode" INTEGER NOT NULL,
        title VARCHAR(100) NOT NULL,
        content TEXT NOT NULL,
        create_time DATETIME,
        "authorId" INTEGER,
        PRIMARY KEY ("postCode"),
        FOREIGN KEY("authorId") REFERENCES student ("studentId")
)
```



```
table|student|student|4|CREATE TABLE student (
        "studentId" INTEGER NOT NULL,
        account VARCHAR(25) NOT NULL,
        pwd VARCHAR(25) NOT NULL,
        PRIMARY KEY ("studentId")
)
table|enrollment|enrollment|5|CREATE TABLE enrollment (
        "studentId" INTEGER,
        "postCode" INTEGER,
        FOREIGN KEY("studentId") REFERENCES student ("studentId"),
        FOREIGN KEY("postCode") REFERENCES post ("postCode")
)
sqlite> _
```

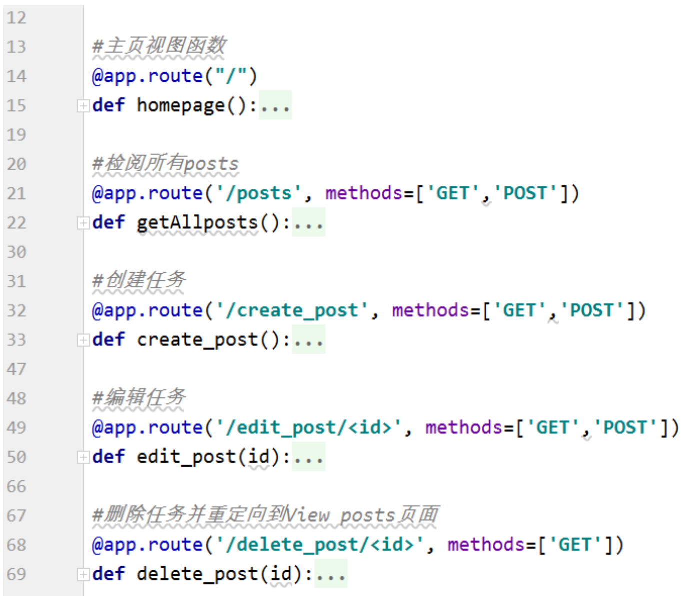
2.4.2 Test if the form takes effect. Test if no data can be successfully submitted.





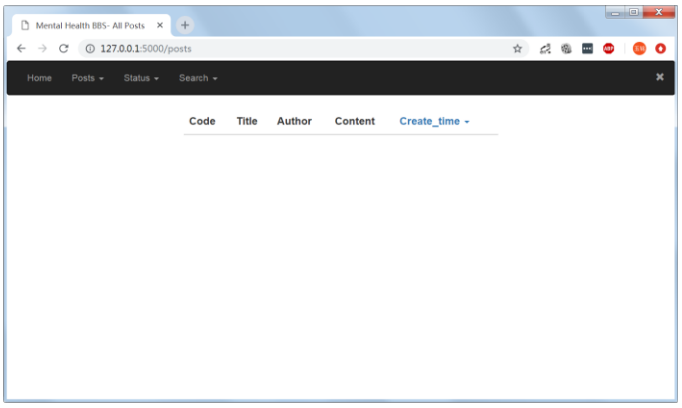
title: Software Engineering

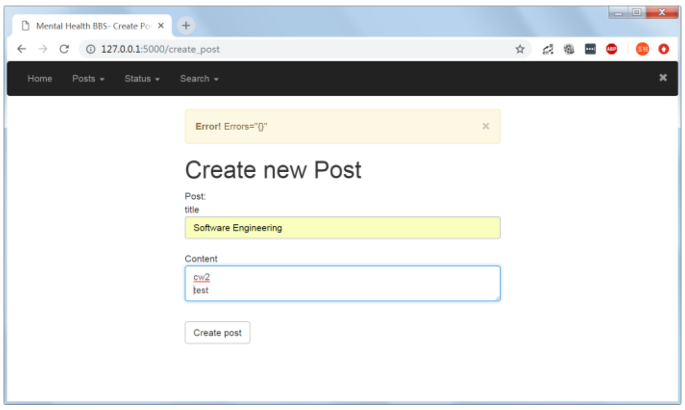
2.4.3 Test each view function to run with the correct logic.



```
#主页视图函数
@app.route("/")
def homepage():...

#检阅所有posts
@app.route('/posts', methods=['GET','POST'])
def getAllposts():...

#创建任务
@app.route('/create_post', methods=['GET','POST'])
def create_post():...

#编辑任务
@app.route('/edit_post/<id>', methods=['GET','POST'])
def edit_post(id):...

#删除任务并重定向到view_posts页面
@app.route('/delete_post/<id>', methods=['GET'])
def delete_post(id):...
```

3 Evidence of deployment



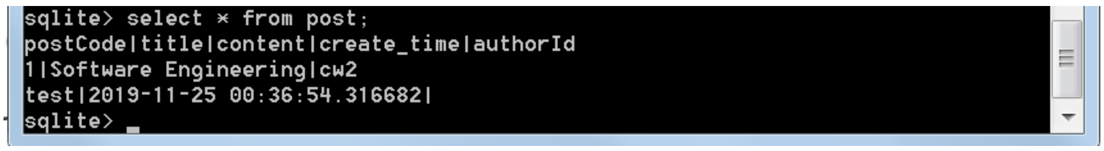


Error! Errors="{}"

title: Software Engineering

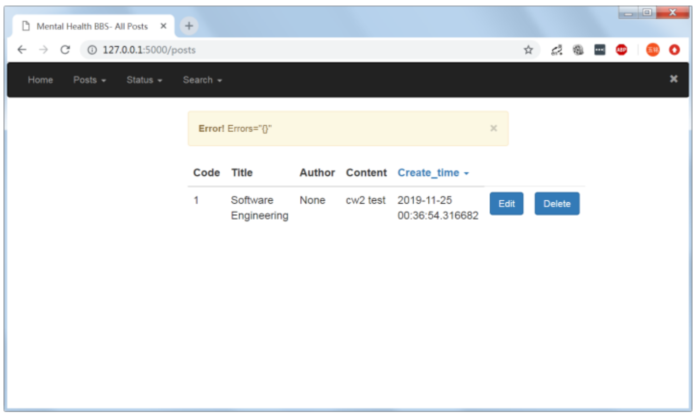
Content: cw2
test

a) Launch the application and check the displayed text is correct; b) Add some data;



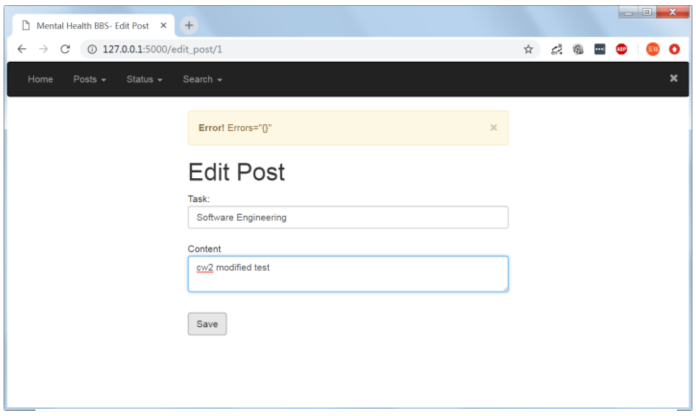
```
sqlite> select * from post;
postCode|title|content|create_time|authorId
1|Software Engineering|cw2
test|2019-11-25 00:36:54.316682|
sqlite> _
```

c) If the data is valid store it on the database;



Error! Errors="{}"

Code	Title	Author	Content	Create_time
1	Software Engineering	None	cw2 test	2019-11-25 00:36:54.316682



Error! Errors="{}"

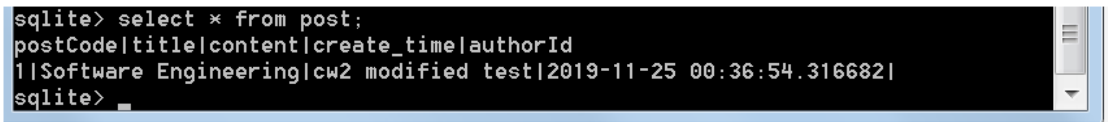
Edit Post

Task: Software Engineering

Content: cw2 modified test

d) Turn off the Application device and re-launch the application;

e) Check the added data is correct and change it to a different value;



```
sqlite> select * from post;
postCode|title|content|create_time|authorId
1|Software Engineering|cw2 modified test|2019-11-25 00:36:54.316682|
sqlite> _
```

f) Check that the changes are stored correctly.

4 Evaluation of design

4.1 Test the technical feasibility of the design

The author designed the most basic architecture for the application. And designed a series of tests to test whether the functions of each basic component can work as expected. This proves the technical feasibility of the project.

4.2 Challenge

The biggest challenge and difficulty of this project is the application of the three-tier architecture. In the flask, the three-tier structure is not completely independent of each other. This brings considerable difficulty to software developers.

4.3 Solution

Iteration and top-down become the solution to the problem.

Requirements Analysis -> Draw Use Case Diagram -> Design Prototype Diagram -> First Generation Simple html Page -> View Function -> Build Relationship Model -> Multiple Iterations

The unresolved issue at present is that the UI layer of the project is too simple and not artistic and aesthetic.