
# COMP9321
# Data Services Engineering
# Assignment 3

# “MovieDigger”
# Project Implementation Document

## Group Name: WeatherSuperman

# Catalog

# 1. Project Preparation Stage(Meeting 1)

## 1.1 Project purpose

The objective of our group assignment is to build a website that can predict the underlying sentiment of a user review of a particular movie. Moreover, this project can also be used for some companies to collect data. Some film production companies can harvest user feedback and preferences through user comments on movies, so as to better improve future movies.

## 1.2 Project Description Statement

“MovieDigger” is a simple web app to analyse the sentiment of a user’s movie review. It allows users to create and log into their own accounts. By inputting one movie review sentence, a user shall receive a binary feedback for the emotion of that review. In order to make the prediction required for this project, machine learning algorithm has been implemented.

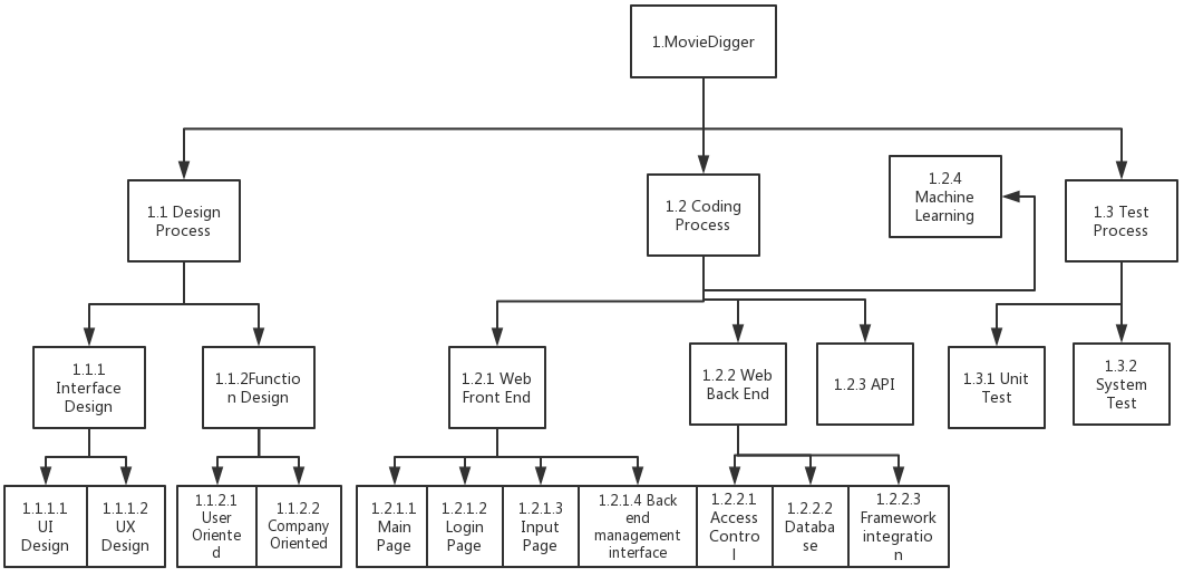
## 1.3 Group Members

| | |
|---|---|
| JIONGQIAN WU (Development Lead) | z5180610 |
| CHUGUAN TIAN | z5145006 |
| ZANLAI HU | z5136463 |
| HONGYIN LIANG | Z5145696 |
| YUNZHUO MA | Z5148910 |

## 1.4 Main Tasks

1. Create a Flask RESTful API with user interfaces.
2. A user is asked to input a movie review within the UI.
3. The Machine Learning model will be trained (we will be testing with quite a few of them, such as Naive Bayes,  SVM, Logistic Regression, Random Forest etc.) and the data used in training process will be stored in MongoDB.
4. The classification/prediction from the ML Model is then sent back to users via the GET function in Restful API.
5. Web_Page-UI will display the result to the user.
6. Debugging(Flask debugging mode) and QA testing will be carried out.

## 1.5 Project  Workflow



## 1.6 Dataset

We will obtain our dataset from the Kaggle https://www.kaggle.com/c/movie-review-sentiment-analysis-kernels-only/data

## 1.7 Communication Channel

Slack Channel
http://9321assignmentiii.slack.com

## 1.8 **Code Repository**

https://github.com/WeatherSuperMan/9321_Group_WeatherSuperMan

# 2. Project Plan

## 2.1 Task Allocation

| Tasks | Allocated to |
|---|---|
| 1.1.1.1 UI Design | YUNZHUO MA |
| 1.1.1.2 UX Design | YUNZHUO MA |
| 1.1.2.1 User Oriented | YUNZHUO MA |
| 1.1.2.2 Company Oriented | YUNZHUO MA |
| 1.2.1.1 Main Page | HONGYIN LIANG |
| 1.2.1.2 Login Page | HONGYIN LIANG |
| 1.2.1.3 Input Page | HONGYIN LIANG |
| 1.2.1.4 Backend management interface | ZANLAI HU, CHUGUAN TIAN |
| 1.2.2.1 Access control | HONGYIN LIANG, CHUGUAN TIAN |
| 1.2.2.2 Database | HONGYIN LIANG |
| 1.2.2.3 Framework Integration | CHUGUAN TIAN, ZANLAI HU |
| 1.3 API | CHUGUAN TIAN, ZANLAI HU |
| 1.2.4 Machine Learning | JIONGQIAN WU |
| 1.3.1 Black Box test | YUNZHUO MA |
| 1.3.2 Usability test | YUNZHUO MA |

## 2.2 Detailed Task Description

| Task | Description |
|---|---|
| 1.1.1.1 UI Design | Based on the principles of easy to understand and easy to use, select the appropriate interface elements, such as text, buttons, text boxes, colors, and so on. As much as possible, the interaction between the user and the interface is simple and efficient.<br>The deliverables are mainly paper models of interface design, line drafts, and low fidelity pictures. |
| 1.1.1.2 UX Design | The process of improving user satisfaction by improving product usability, ease of use. The main task is to conduct research before the project starts and conduct usability testing of products after the project goes online. |
| 1.1.2.1 User Oriented | This project is mainly for individual users. It is a website that can predict the emotions of user input and can be used as an online forecasting website.<br>Deliverables are functional and non-functional requirements document |
| 1.1.2.2 Company Oriented | This project can also be used for some companies to collect data. Some film production companies can collect user feedback and preferences through user comments on movies, so as to better improve future movies.<br>Deliverables are functional and non-functional requirements document |
| 1.2.1.1 Main Page | html5,css3,js implement what UI design |
| 1.2.1.2 Login Page | html5,css3,js implement what UI design |
| 1.2.1.3 Input Page | html5,css3,js implement what UI design |
| 1.2.1.4 Backend management interface | implement the restful api UI via the swagger structure |
| 1.2.2.1 Access control | python flask and HTML connection |
| 1.2.2.2 Database | PyMongo,mlab and python flask connection,return login,register information |
| 1.2.2.3 Framework Integration | implement the ML model into app use |

| | |
|---|---|
| 1.3 API | build the restful api and wait to be connected to UI |
| 1.2.4 Machine Learning | The machine learning model used here is Complement Naive Bayes, which is an efficient algorithm introduced in the latest version(0.20)) of Scikit-learn module. An average accuracy of 94% was achieved! |
| 1.3.1 Black Box test | Tested at the program interface without considering the internal structure and internal characteristics of the program at all, it only checks whether the program function is normally used as specified in the specification specification, and whether the program can properly receive the input data to produce correct output information.<br>Mainly trying to find the following errors<br>Incorrect function or omission;<br>Interface error;<br>Input and output errors;<br>Database access error;<br>Performance error;<br>Initialize and terminate errors, etc.<br>The deliverable is a test evaluation |
| 1.3.2 Usability test | The usability test includes testing for the application and also includes testing the user manual system documentation.<br>Quality external models are often used to evaluate ease of use. Includes tests in the following areas:<br>(1) Comprehensibility test;<br>(2) Easy to learn test;<br>(3) Easy operation test;<br>(4) Attraction test;<br>(5) Easy-to-use compliance test.<br>This kind of test mainly considers the user experience, so it will focus on the interface test.<br>The deliverable is a overview evaluation of usability test |

## 2.3 Planning Schedule

| Word Packages | Start Date | End Date |
|---|---|---|
| 1.1.1.1 UI Design | 2018/10/05 | 2018/10/10 |
| 1.1.1.2 UX Design | 2018/10/05 | 2018/10/10 |
| 1.1.2.1 User Oriented | 2018/10/17 | 2018/10/19 |
| 1.1.2.2 Company Oriented | 2018/10/17 | 2018/10/19 |
| 1.2.1.1 Main Page | 2018/10/06 | 2018/10/07 |
| 1.2.1.2 Login Page | 2018/10/08 | 2018/10/09 |
| 1.2.1.3 Input Page | 2018/10/10 | 2018/10/11 |
| 1.2.1.4 Back end management interface | 2018/10/11 | 2018/10/13 |
| 1.2.2.1Access control | 2018/10/17 | 2018/10/19 |
| 1.2.2.2 Database | 2018/10/11 | 2018/10/13 |
| 1.3 API | 2018/10/08 | 2018/10/12 |
| 1.2.4 Machine Learning | 2018/10/06 | 2018/10/17 |
| 1.3.1 Unit test | 2018/10/17 | 2018/10/19 |
| 1.3.2 System test | 2018/10/17 | 2018/10/19 |

# 3. Project Implementation

## 3.1 Design Process

The design of the UI interface will focus on color matching and typesetting for a more optimized experience. Some cute icons and buttons are designed to engage the user. (Figure 1).

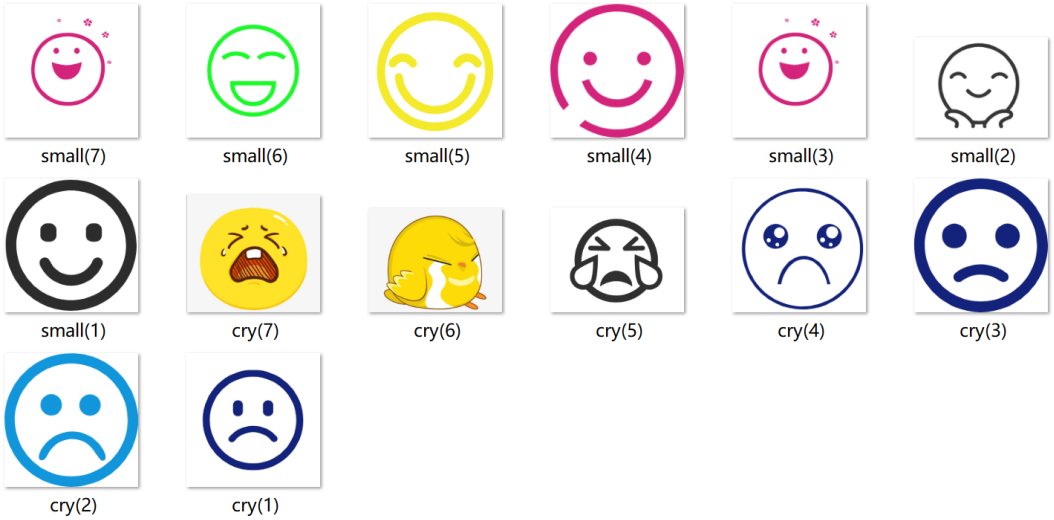



Figure 1: Pre-design about emoji

Logo is extremely important in the design of web pages. Based on design principles of simple, clear, and understandable, here is the preliminary design of my logo for the web app. (Figure 2)


Figure 2: Logo of website

## 3.2 Coding Process

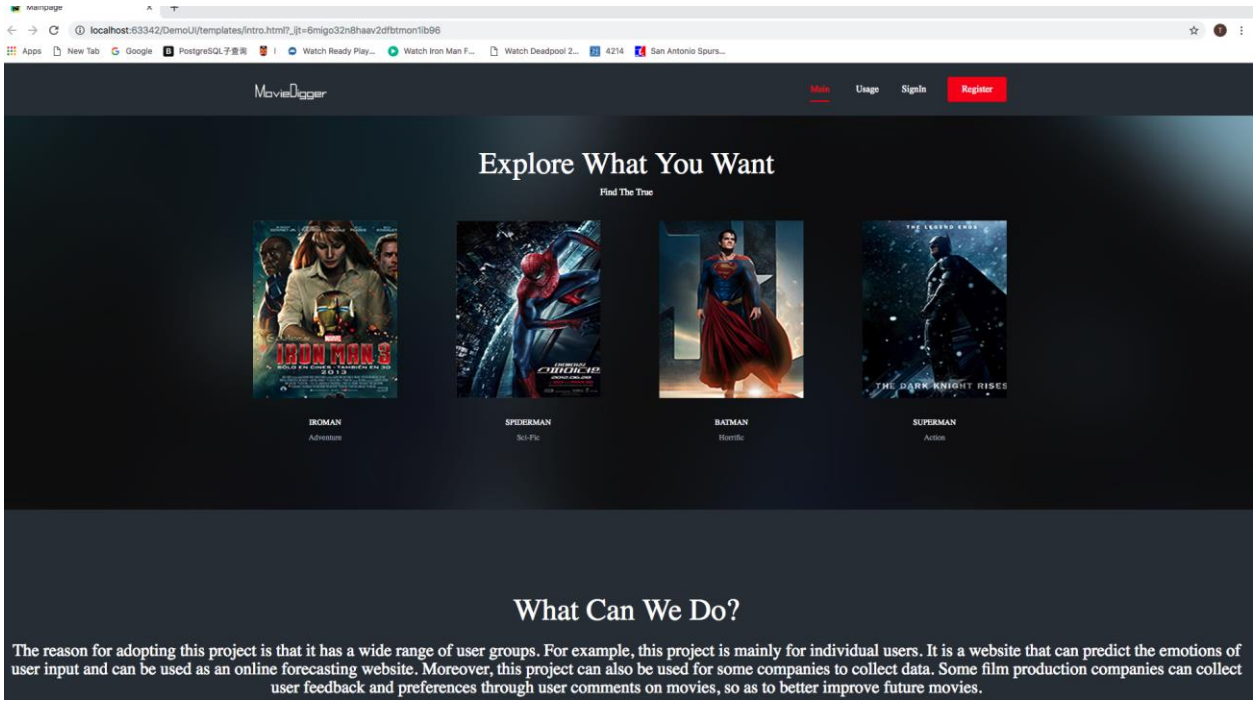
### 3.2.1 UI Implementation and service design,database

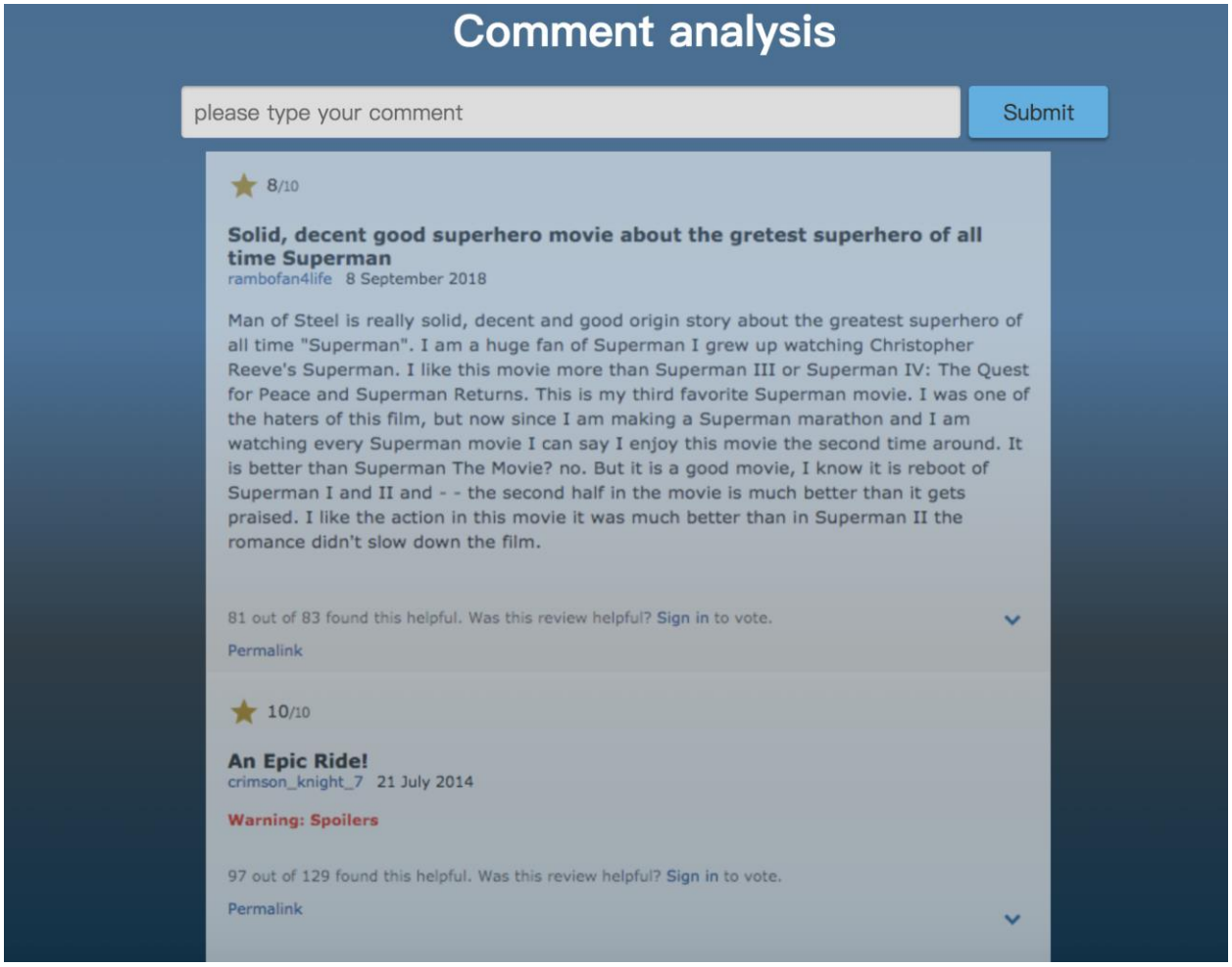
| | |
|---|---|
| UI Structure | HTML5 + CSS3 + JAVASCRIPT |
| Service | Python Flask |
| Database | Pymongo |

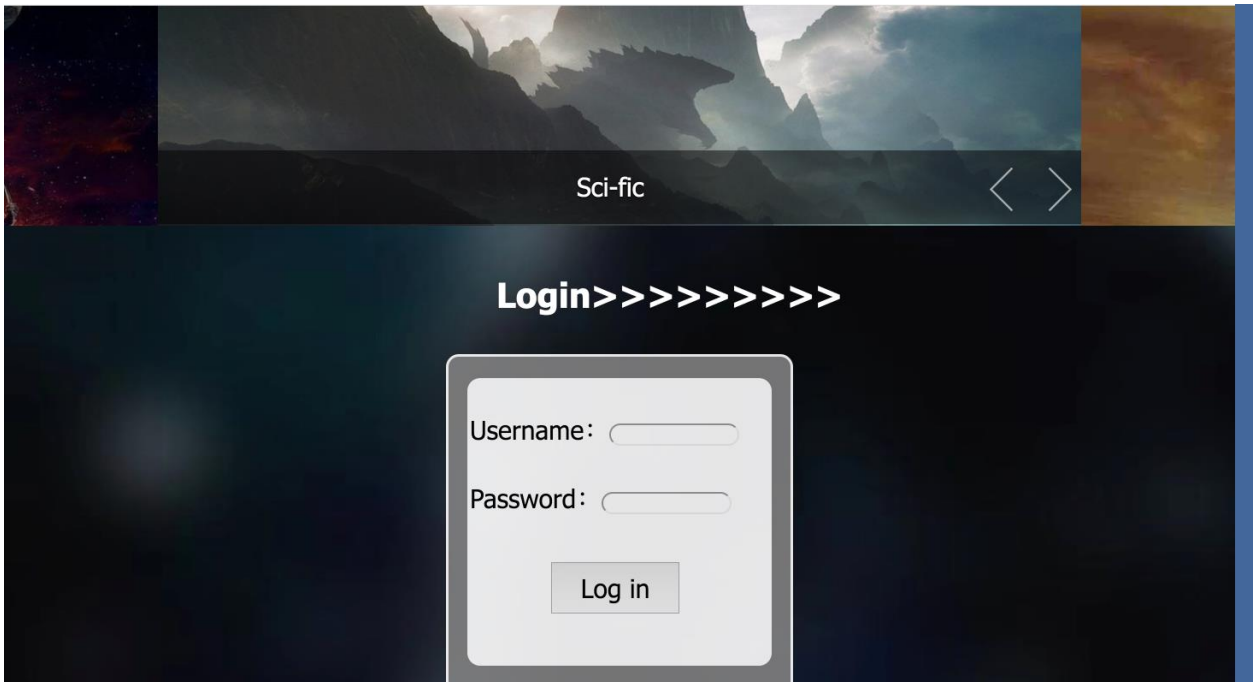
- HTML Form label transmit data to Flask service,

```
<div class ="bg_login">
  <form method="get"  class="login_form" onsubmit="return check()" action="http://localhost:5000/handle_register">
    Username: <input type="text" name="Username" id = 'use' class="login_text"/></br>
    Password: <input type="password" name="Password" id = 'reg' class="login_text"/></br>
    <input type="submit" value="Register" id="LOG" class="submit_sty">
  </form>
  <form action="login.html">
    <input type="submit" value="cancel" id="REG" class="regester_sty" >
  </form>
```

- javascript implement HTML animation(slide,autoplay)


MovieDigger
Main
Usage
SignIn
Register
Explore What You Want
Find The True
IRONMAN
Adventure
SPIDERMAN
Sci-Fic
BATMAN
Horrific
SUPERMAN
Action
What Can We Do?
The reason for adopting this project is that it has a wide range of user groups. For example, this project is mainly for individual users. It is a website that can predict the emotions of user input and can be used as an online forecasting website. Moreover, this project can also be used for some companies to collect data. Some film production companies can collect user feedback and preferences through user comments on movies, so as to better improve future movies.


Comment analysis
please type your comment
Submit
8/10
Solid, decent good superhero movie about the gretest superhero of all time Superman
rambofan4life 8 September 2018
Man of Steel is really solid, decent and good origin story about the greatest superhero of all time "Superman". I am a huge fan of Superman I grew up watching Christopher Reeve's Superman. I like this movie more than Superman III or Superman IV: The Quest for Peace and Superman Returns. This is my third favorite Superman movie. I was one of the haters of this film, but now since I am making a Superman marathon and I am watching every Superman movie I can say I enjoy this movie the second time around. It is better than Superman The Movie? no. But it is a good movie, I know it is reboot of Superman I and II and - - the second half in the movie is much better than it gets praised. I like the action in this movie it was much better than in Superman II the romance didn't slow down the film.
81 out of 83 found this helpful. Was this review helpful? Sign in to vote.
Permalink
10/10
An Epic Ride!
crimson_knight_7 21 July 2014
Warning: Spoilers
97 out of 129 found this helpful. Was this review helpful? Sign in to vote.
Permalink




- HTML form send data to service,and connect to database

```
app = Flask(__name__)
connection = pymongo.MongoClient('ds255282.mlab.com', 55282)
db = connection['assignment']
db.authenticate('z5145696', 'Magic8484')
account = db['account']
```

### 3.2.2 Machine Learning

Machine Learning Structure:

- Raw training data (.tsv) is converted into dataframe using panda.
- We are interested in the Sentiment section of the dataset, therefore this particular column will be the independent variable.
- The sentiment score ranges from 0 to 4, however, only the two extremes will be considered as binary outcome is what we are aiming for. Hence, an extra column of binary outcome is added to the dataframe after all rows with a sentiment score of 1,2,3 is discarded.
- Take advantage of the TfidVectorizer from scikit learn feature extraction library to convert the text into a matrix of TF-IDF(frequency–inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection.) in two steps:
  i) Fit the TF-IDF Vectorizer to the text
  ii) Transform the text data to a sparse TF-IDF matrix
- Set the binary column (0 for negative, 1 for positive) as the dependent variable.
- Now that we have gathered both independent and dependent variables, we split them into 4 sets (X_train, X_test, y_train, y_test), the size of the test set is set to 20% of the original training set.
- Convert both X_train, X_test into dense matrices. (this step is optional)
- Complement Naive Bayes was implemented as it gave the best accuracy
  - Here is a research paper on the topic
  - https://people.csail.mit.edu/jrennie/papers/icml03-nb.pdf
  -
- The trained model and vectorizer were saved into two pickle files for later use.
- Machine Learning Part Complete!!

All training dataset was converted into dense matrix before training for fair comparison!

| Trial Run | T | LR | T | MN_NB | T | G_NB | T | B_NB | T | C_NB |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 460 | 0.9330 | 510 | 0.9349 | 2490 | 0.9186 | 1860 | 0.9235 | 500 | 0.9413 |
| 2 | 470 | 0.9346 | 490 | 0.9294 | 2600 | 0.9248 | 1860 | 0.9186 | 490 | 0.9398 |
| 3 | 460 | 0.9269 | 490 | 0.9380 | 2590 | 0.9226 | 1850 | 0.9180 | 490 | 0.9392 |
| 4 | 460 | 0.9370 | 490 | 0.9330 | 2530 | 0.9284 | 1900 | 0.9284 | 490 | 0.9429 |
| 5 | 450 | 0.9266 | 510 | 0.9241 | 2570 | 0.9131 | 1910 | 0.9220 | 490 | 0.9420 |
| 6 | 450 | 0.9312 | 500 | 0.9340 | 2560 | 0.9220 | 1870 | 0.9158 | 510 | 0.9472 |
| 7 | 460 | 0.9318 | 500 | 0.9343 | 2790 | 0.9269 | 1850 | 0.9229 | 500 | 0.9407 |
| 8 | 450 | 0.9380 | 500 | 0.9241 | 2680 | 0.9226 | 1860 | 0.9158 | 490 | 0.9429 |
| 9 | 440 | 0.9217 | 500 | 0.9426 | 2670 | 0.9165 | 1890 | 0.9171 | 490 | 0.9389 |
| 10 | 450 | 0.9294 | 500 | 0.9306 | 2600 | 0.9291 | 1870 | 0.9140 | 500 | 0.9361 |
| **Mean Value** | **455** | **0.9310** | **499** | **0.9325** | **2608** | **0.9225** | **1872** | **0.9196** | **495** | **0.9411** |

**The winner is Complement Naive Bayes!!**
https://people.csail.mit.edu/jrennie/papers/icml03-nb.pdf

T = Training Time (milliseconds)
LR = Logistic Regression
MN_NB = Multinomial Naïve Bayes
G_BN = Gaussian Naïve Bayes
B_NB = Bernoulli Naïve Bayes
C_NB = Complement Naive Bayes

I have set a maximum training time of 30 seconds, any ML algorithm that takes more than 30 seconds to train the dataset is deemed to have failed.

Failed ML algorithm:

Linear Regression
Support Vector Machine
Decision Tree

### 3.2.3 Backend management interface

For this part, after we build the restful api, then the swagger UI is automatically generated.

### 3.2.4 Access control

For this part, we use python to create connection to html file and connect the function in python to be used in the webpage.

### 3.2.5 Framework Integration

We integrate the ML model to the api, and make it to be used and get the output. Apart from this, we also integrate all the changed files to be used, test and modify.

### 3.2.6 API

For this part, we build a flask-restful server api and create some functions to get access to the Webpage-ui and make the model of machine learning into use.

## 3.3 Test Process

We found a total of four tester for usability test. The first tester is a graduate student major in art and design in UNSW, Because we want to get some professional interface design suggestions. The second one is an apartment DBA in one Chinese company, because we want to get more professional and specific improvement suggestions. The other

two testers are both ordinary students and they don't have relevant knowledges on IT or Design. I think it was great to have someone unbiased to pinpoint the good and bad things about our project.

Testers were asked to complete three tasks without any hints.

***Tasks:***

1. ***Login or register an account***
2. ***Input some words you like***
3. ***Receive feedback***

These 3 tasks mainly focus on whether the user understand our web site; Users are able to complete all the operation independently or not; Whether the user interface are satisfied.

In order to verify the accuracy of machine learning results, Black box testing is used and 14 test cases were used to determine the prediction accuracy of the machine learning model.

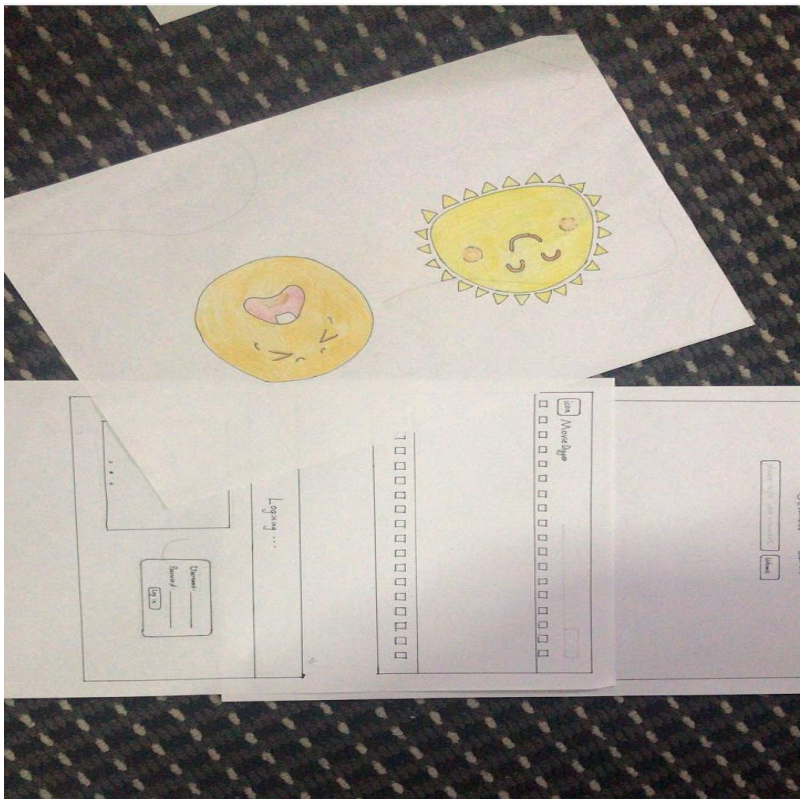
# 4. Deliverables display

## 4.1 Design Process

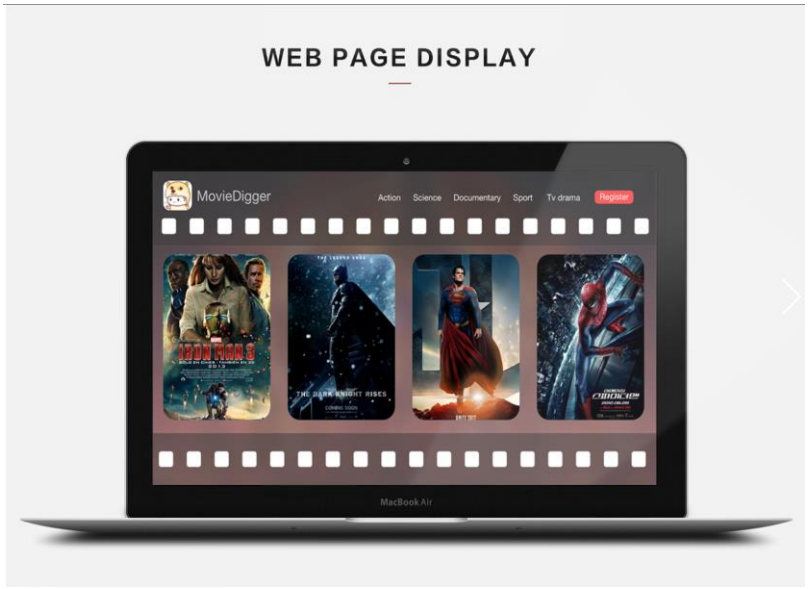
### requirement analysis

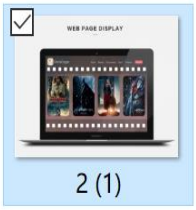
Functional requirements:
Allow users to log in or register a new account
Allow users to enter text, and the output is the emotional result of the text

Non-Functional requirements
Support movie poster scrolling
The output is a cute emoji



Prototype of UI design



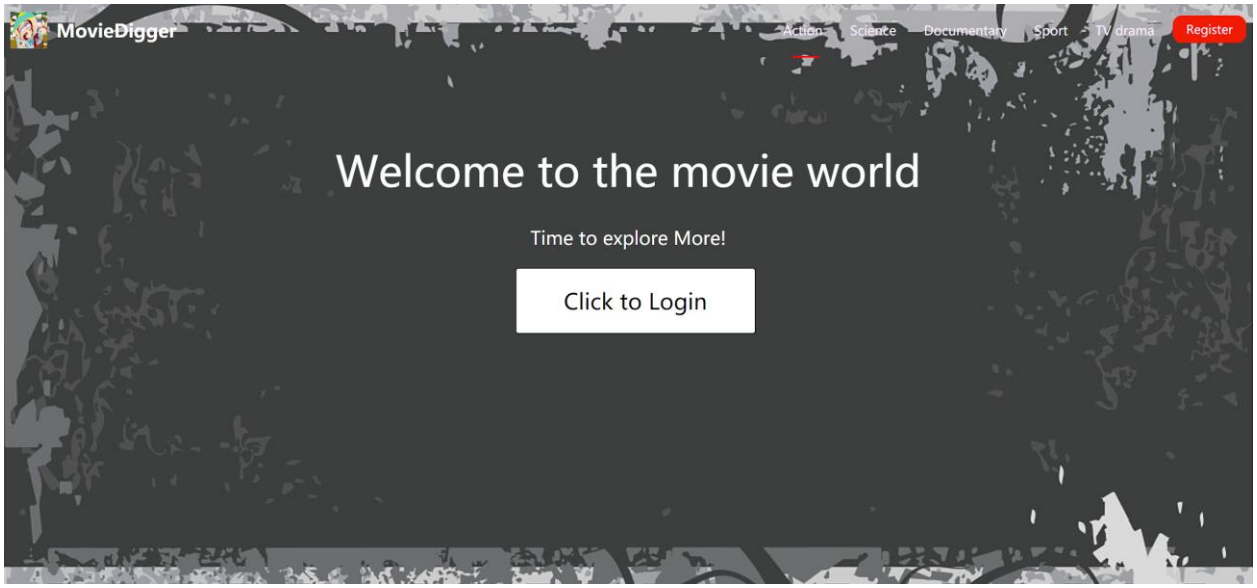





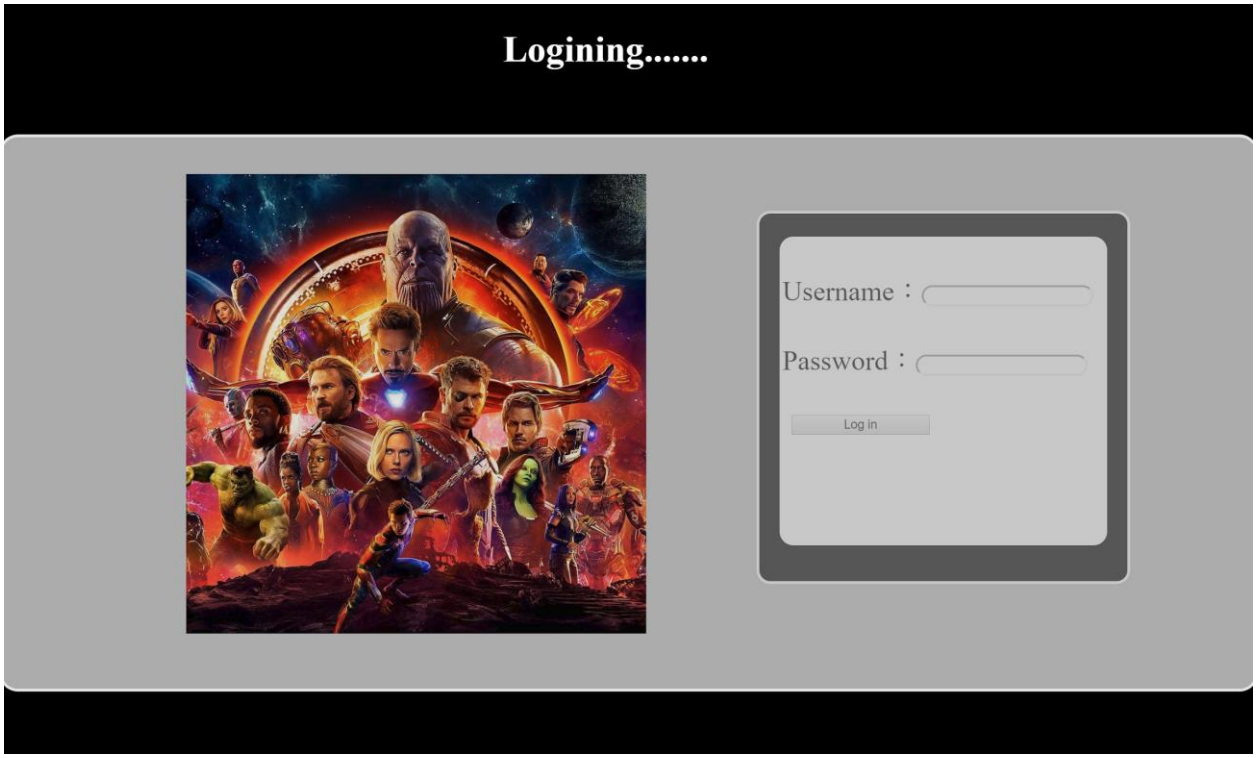




low fidelity pictures




Actual homepage




actual login page


search page

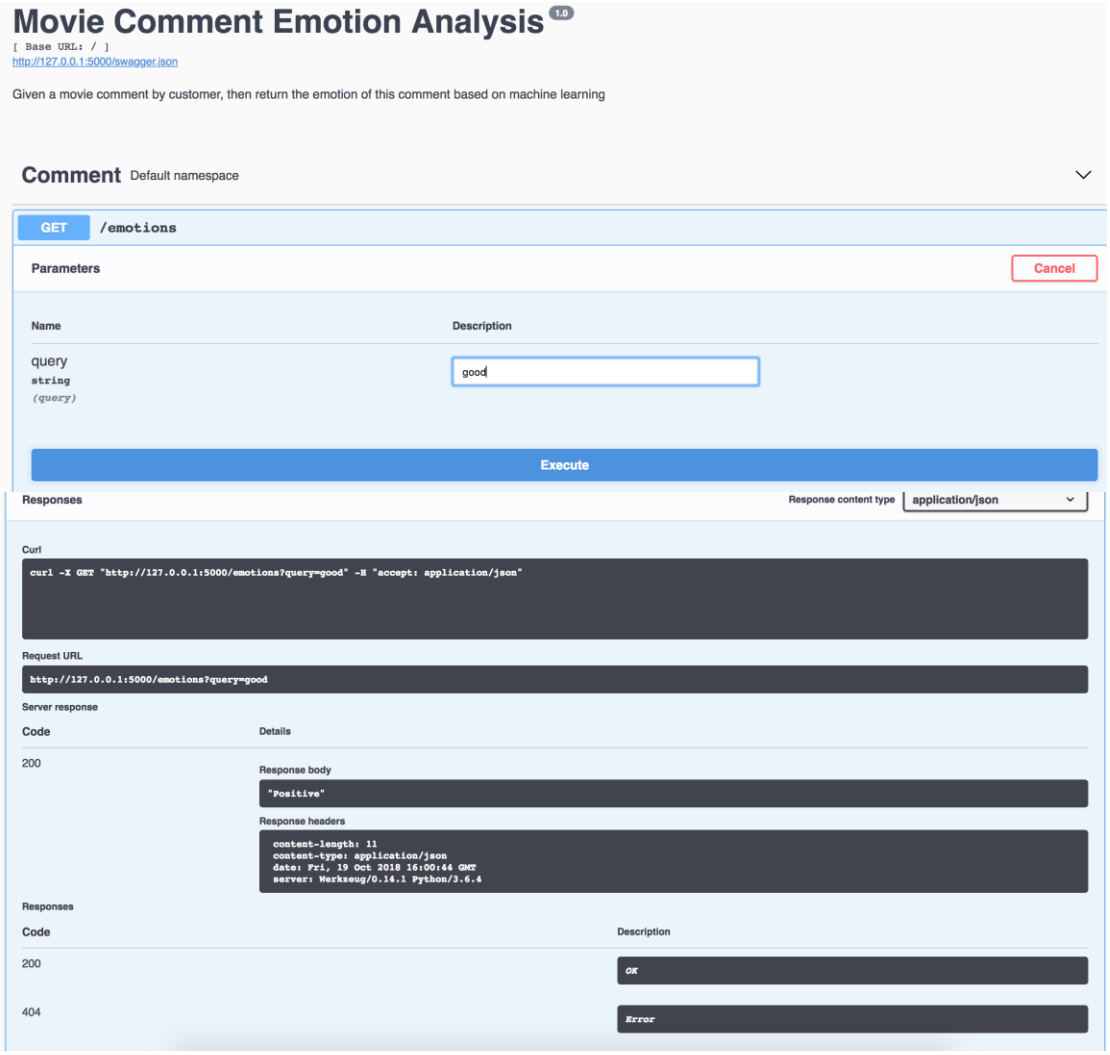
## 4.2 Coding Procedures

### Machine Learning Backend

```
# ML imports
from sklearn.naive_bayes import ComplementNB
from sklearn.feature_extraction.text import TfidfVectorizer

import pickle


def build_model():
    model = NLPModel()
    with open('lib/data/train.tsv') as f:
        data = pd.read_csv(f, sep='\t')

    pos_neg = data[(data['Sentiment'] == 0) | (data['Sentiment'] == 4)]
    pos_neg['Binary'] = pos_neg.apply(
        lambda x: 0 if x['Sentiment'] == 0 else 1, axis=1)

    model.vectorizer_fit(pos_neg.loc[:, 'Phrase'])
    print('Vectorizer fit complete')

    X = model.vectorizer_transform(pos_neg.loc[:, 'Phrase'])
    print('Vectorizer transform complete')
    y = pos_neg.loc[:, 'Binary']

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)

    # convert to dense matrix
    X_train = X_train.toarray()
    X_test = X_test.toarray()

    model.train(X_train, y_train)
    print('Model training complete')

    model.pickle_clf()
    model.pickle_vectorizer()
```

```python
class NLPModel(object):

    def __init__(self):

        self.clf = ComplementNB()
        self.vectorizer = TfidfVectorizer()

    def vectorizer_fit(self, X):

        self.vectorizer.fit(X)

    def vectorizer_transform(self, X):

        X_transformed = self.vectorizer.transform(X)
        return X_transformed

    def train(self, X, y):

        # X_train, X_test, y_train, y_test = train_test_split(X, y)
        self.clf.fit(X, y)

    def predict_proba(self, X):
        """Returns probability for the binary class '1' in a numpy array
        """
        y_proba = self.clf.predict_proba(X)
        return y_proba[:, 1]

    def predict(self, X):
        """Returns the predicted class in an array
        """
        y_pred = self.clf.predict(X)
        return y_pred

    def pickle_vectorizer(self, path='lib/models/TFIDFVectorizer.pkl'):
        """Saves the trained vectorizer for future use.
        """
        with open(path, 'wb') as f:
            pickle.dump(self.vectorizer, f)
            print("Pickled vectorizer at {}".format(path))

    def pickle_clf(self, path='lib/models/SentimentClassifier.pkl'):
        """Saves the trained classifier for future use.
        """
        with open(path, 'wb') as f:
            pickle.dump(self.clf, f)
            print("Pickled classifier at {}".format(path))
```

## 4.2.1 Backend management interface



## 4.2.2 Access control

```
@app.route('/handle_login', methods = ['POST', 'GET'])
def handle_login():
    if request.method == 'POST':
        return render_template("login.html")
    if request.method == 'GET':
        user = request.args.get('Username')
        password = request.args.get('Password')
        for i in account.find():
            if user in i['username'] and password in i['password']:
                return render_template("search.html")
            else:
                continue
    return u'password or username error'
```

```
<div class ="bg_login">
    <form method="post" action="search.html" class="login_form">
        Username: <input type="text" name="Username" class="login_text"/></br>
        Password: <input type="password" name="Password" class="login_text"/></br>
        <input type="submit" value="Log in" id="" class="submit_sty">
        <input type="submit" value="Regester" id="" class="regester_sty">
    </form>
    </div>
</div>
```

```
@app.route('/handle_register', methods = ['POST', 'GET'])
def handle_register():
    mydict = {}
    if request.method == 'POST':
        return render_template("intro.html")
    if request.method == 'GET':
        user = request.args.get('Username')
        password = request.args.get('Password')
        # user = request.args.get('Username')
        # password = request.args.get('Password')
        # return u'regester success'
        mydict = {"username": user, "password": password}
        account.insert_one(mydict)
        for i in account.find():
            if user in i['username'] and password in i['password']:
                return u'password already exists'
            else:
                continue
        return render_template('login.html')
```

*Define the same function both in html and python, when implement the function, return the html page.*

### 4.2.3 Framework Integration

```
model = NLPModel()

clf_path = 'lib/models/SentimentClassifier.pkl'
with open(clf_path, 'rb') as f:
    model.clf = pickle.load(f)

vec_path = 'lib/models/TFIDFVectorizer.pkl'
with open(vec_path, 'rb') as f:
    model.vectorizer = pickle.load(f)

user_query = query
uq_vectorized = model.vectorizer_transform(np.array([user_query]))
prediction = model.predict(uq_vectorized)

int(prediction)
pred_proba = model.predict_proba(uq_vectorized)

confidence = round(pred_proba[0], 3)
```

*implement the ML model to api to be used*

### 4.2.4 API

*For this part, we build a api just like we learned in the assignment 2.*

```
app = Flask(__name__)
app.config.from_object(config)

api = Api(app,
        default="Comment",  # Default namespace
        title="Movie Comment Emotion Analysis",  # Documentation Title
        description="Given a movie comment by customer, then return the emotion of this comment based on machine lear

@app.route('/handle_login', methods = ['POST', 'GET'])
def handle_login():
@app.route('/handle_register', methods = ['POST', 'GET'])
def handle_register():
@app.route('/handle_data', methods = ['POST', 'GET'])
def handle_data():
```

## 4.3 User Testing Procedures

# ISSUE TABLE ABOUT THE MOVIE COMMENT PREDICTION MACHINE

| Issue no. | Task | Issue | Principle | Type | Severity (1-10) |
|---|---|---|---|---|---|
| 1 | 3 | • User satisfied with the feedback when typing the words | User goal Utility | + | 9 |
| 2 | 2 | • User easily understand all symbol meanings | Design principle Affordance | + | 9 |
| 3 | 1 | The user login and registration are very convenient | Design principle Mapping | + | 9 |
| 4 | 2 | Words kinds of small | Design principle Visual | - | 7 |

| | | | | | |
|---|---|---|---|---|---|
| 5 | 3 | Happy with the icon design | Design principle Visual | + | 9 |
| 6 | 1 | No help document | Usability goals Learnability | - | 5 |
| 7 | 1 | Good error feedback | Usability goals safety | + | 9 |

## User Evaluation Form

| | USER 1 | USER 2 | USER3 | USER 4 |
|---|---|---|---|---|
| easy to read | 7 | 4 | 8 | 10 |
| visually appealing | 8 | 8 | 7 | 7 |
| well designed | 8 | 7 | 9 | 10 |
| friendly with new user | 9 | 9 | 10 | 9 |
| my mistakes were easy to correct | 9 | 9 | 8 | 9 |
| compared to expected, tasks can be achieved smoothly | 10 | 10 | 9 | 10 |
| easy to find specific information | 8 | 7 | 6 | 8 |

## Black Box Test Evaluation

## A few random test cases from Kaggle Test dataset

| Test case | Expected output | Actual output | Correct(Y) / Incorrect(N) |
|---|---|---|---|
| kids will go happily along for the ride . | positive | positive | Y |
| sparklingly inventive and artful , always fast and furious tale | positive | positive | Y |
| make Deutchland a popular destination for hungry tourists | positive | positive | Y |
| is alive and well and living in LA | positive | positive | Y |
| What 's surprising about Full Frontal is that despite its overt self-awareness , parts of the movie still manage to break past the artifice and thoroughly engage you . | positive | positive | Y |
| horribly mediocre | negative | negative | Y |
| The sad thing about Knockaround Guys | negative | negative | Y |

| | | | |
|---|---|---|---|
| some awful acting and lame special effects | negative | negative | Y |
| So unremittingly awful that labeling it a dog probably constitutes cruelty to canines . | negative | negative | Y |
| It 's so mediocre , despite the dynamic duo on the marquee , that we just can't get no satisfaction . | negative | negative | Y |

# 5.Conclusion

The project was implemented successfully as it was able to pass more than 90% of test cases while remaining stable, and easy to maintain. It has, without a doubt, been a fantastic learning opportunity for each and every member of the team. In spite of numerous technical and communication challenges, we have been able to overcome all obstacles to create this little masterpiece of our own. The experience we have gained will greatly enhance essential skills pertaining to collaboration, communication and time management in addition to boosted confidence in coding.