

Project Portfolio

Group 38

Qiupu Shen, Wencheng Zhang, Xiaohang Tang, Yixing Lu, Yongyin Yang,
Yunfan Shi

Report

Contents

1 People and Roles	1
2 Application Overview	3
3 Achievements	4
4 Evaluation	5
5 Future developments.....	8
6 Professional issues	9
7 Reference.....	10

1 People and Roles

Sub-Teams	Name	Achievements	Abilities
Front-End	Yixing Lu	<ul style="list-style-type: none">• Designed and initiated system architecture• Implemented user history and report download function• Managed technical development using Git version control	<ul style="list-style-type: none">• Problem-solving• Effective communication in teamwork• Strong self-learning skills
	Wencheng Zhang	<ul style="list-style-type: none">• Delving more jQuery• Learning of Vue	<ul style="list-style-type: none">• Better communication with

		architecture and the life cycle of its components & <ul style="list-style-type: none"> • Regular expression • ECharts and data dimensions • Transfer and filter data set 	group members <ul style="list-style-type: none"> • Knowledge of Vue architecture.
Back-End	Yunfan Shi	<ul style="list-style-type: none"> • DB query, update & migrate • Parameter/data passing with frontend • Demand curve • RFM • LSTM framework • Algorithm Django integration • Environment configuration • Deploy website • Server maintenance 	<ul style="list-style-type: none"> • Effective communication • Problem-solving • Pressure handling • Learning by doing • Self-study
	Xiaohang Tang	<ul style="list-style-type: none"> • DB query & update & migrate • User account & history & report download • Holt-winters Model implementation 	<ul style="list-style-type: none"> • Efficient communication • Problem-solving & error detecting
	Qiupu Shen	<ul style="list-style-type: none"> • General Scheduling and Management • LSTM Model Training and Optimisation • Optimising Algorithm Design 	<ul style="list-style-type: none"> • Efficient communication and teamwork • Divergent Thinking • Error Detecting and Critical Thinking
Test	Yongyin Yang	<ul style="list-style-type: none"> • Unit Testing, Integration Testing & System Testing & Acceptance Testing • Documentation • Management 	<ul style="list-style-type: none"> • Efficient communication • Error correction

2 Application Overview

Our project "Area38" is a lightweight and AI-based Business Intelligence Analysis Website which provides commercial entities, especially individual merchants, a handy platform where they can obtain their own business metrics forecast based on their historical sales data. Specifically, we expect to provide users with Demand curve fitting for optimised pricing strategies, RFM customer segmentation and AI/statistical time-series prediction to help them achieve a higher profit or customised goals.

The website can facilitate businesses to improve their data-driven decision making in a user-friendly way without seeing complex code and numbers with an administrative system for managers. But the scope is limited to some extent, and the algorithm is not flexible enough to provide comprehensive advice for better reference.

The application has three types of users: Guests, Registered Users and Administrators. Guests and Registered Users cannot access Admin functions. LSTM and Holt-Winters are Registered Users only. Admin can only monitor user accounts and update the LSTM model.

Possible queries & data processing features are listed below:

Feature	Related Query (DB only)	Data Processing
register, login/out	select from UserInfo table	cookies update, pack the data from user into FormData and transmit to backend
four Core functions' analysis result	None	upload/ file IO result/history store, receive data from backend and convert it into JSON data structure
demand curve	None	curve fitting, convert JSON data to two dimensions data points
RFM	None	input data Customer group aggregation, convert JSON to data groups
time-series	None	load pre-trained model, input data split, transform, convert JSON data to two dimensions data points
history	select from UserFile/UserLog table	None
pdf download	select from UserLog table	read analysis result image, read user actions
upload	select from UserInfo table	get current user name, store file to the server directory with username
admin:	select from UserInfo	replace LSTM pretrained model

3 Achievements

In this section, we will explain our achievements compared to the agreed requirements.

3.1 Basic user services

- User account services: User account services, including register, login/logout (cookie-based) and user history (DB based), are implemented. Users can see their history operations (including date, action and uploaded filename) on the history page.
- File management: We implemented a file management mechanism for all the data processing on the website. Users can choose a local file and upload it to our website. The files are saved in server native directories in the name of each user. They can also reset and choose a new file to upload. Each function would index the corresponding file in the same manner.

3.2 Core functions

For our three main functions, we implemented four models and employed three different types of charts to visualise the results. We also include boxplots for all columns of the uploaded data, which provides users with an overview of their raw data. Finally, we generate a PDF analysis report based on fixed templates and analysis history for users' reference.

- Demand Curve:

We implemented a fixed mathematical model with two parameters and used SciPy to conduct demand curve fitting. We also implemented a slide bar next to the demand curve chart. Users can move the slider and see a real-time update of their revenue w.r.t the target price.

- Customer group analysis:

We implemented RFM customer segmentation algorithm based on prior knowledge and database aggregation functions, the result of which is concisely visualised in a Treemap.

- Time-series prediction:

We trained, fine-tuned and deployed an LSTM model for time-series prediction. The design of this function is considered the highlight of our project (more details in section 3.4). In addition, we have implemented an extra statistical model (Holt-Winters) to cross-validate. We will have a critical evaluation of the two models in Evaluation section 4. We added more interaction to the chart, where users can adjust the bi-directional slider to see their period of interest for time-series prediction.

3.3 System Management

We implemented a basic admin page allowing services to be monitored and updated. As for monitoring, only user account information monitoring is

implemented. In addition, Administrators can easily update the LSTM model via a file uploading mechanism.

3.4 Project Highlight

A highlight of our project is that we have improved the LSTM based function for time-series prediction. Users can customise how many days in the future to predict their business metrics.

In most practices, the prediction is usually based on test data, the ground truth of which is already known. Hence, it is considered less useful. In our implementation, we encode it as a parameter and let the user decide how far they want to predict. This essentially adds value and flexibility to the model while improving the interaction between users and our website.

4 Evaluation

4.1 Evaluation Strategy

According to existing research (CIPD, 2020) and book (Doyle, 2011), we customised the evaluation strategy by integrating SWOT/Value analysis as well as a user survey. In addition, we also evaluated by comparing actual project deliverables with our initial requirement and quality inspection. Based on statements in the Achievements section and objectives stated in the requirement analysis, overall core functions are considered 70% complete while the rest are 60% complete.

4.2 Validation by UEQ-S user survey

We invited ten participants (including one female) with an average age of 20.2 (s.d. = 8.11) to use our website and complete the User Experience Questionnaire Short Version (UEQ-S) (Schrepp, Thomaschewski and Hinderks, 2017). We get the mean overall score of 1.89 (s.d. = .80), mean Pragmatic Quality Score of 2.10 (s.d. = .75), and mean Hedonic Quality score of 1.68 (s.d. = .92), where scores' range is (-3 ~ 3). All of our scores achieve the excellent benchmark of UEQ-S. Moreover, more than 75% of respondents provided positive feedback in terms of the website's user experience; only in a few questions did there exist 1-2 neutral replies. This could, to some extent, validate that the actual user experience of the implemented website could generally meet the aim and objectives, especially in terms of user-friendliness and clarity.

4.3 Evaluation Conclusion

This project aims to build a lightweight, all-visualised business intelligence website to provide users with pricing and marketing strategies as well as business metric prediction. Based on the project objectives stated in the Requirement analysis, the evaluation is as follows:

1. Fully operational register, login/log out; change password function has only web interface; delete account data web interface, and function is not implemented
2. Basic demand curve fitting using a fixed mathematical model based on user input; advice for pricing strategy is only limited to predicting potential revenue given a user input price. The real-time update is only limited to the price and corresponding revenue when the user moves the scrollbar.
3. RFM customer grouping algorithm with basic statistics integration, file IO. It provides customer group segmentation based on a fixed database aggregation algorithm and fixed RFM group value threshold.
4. Time series prediction only via fixed architecture pre-trained LSTM model in server. In addition, the user could adjust the time axis to see the local detail of the curve, but this comes from the frontend chart framework: ECharts (ECharts.apache.org, n.d.). Result dashboard is not implemented, only direct result from LSTM or Holt-winters.
5. User upload file from local computer is fully implemented, but fetching files from user's cloud drive is not implemented
6. Basic analysis report is generated: put backend generated image into PDF report template and allow users to download
7. Implemented service monitoring for admin by querying user info in DB and sending to frontend to display, but no DB transaction monitoring is implemented
8. Only the LSTM model can be easily updated/upgraded by uploading and replacing the pre-trained LSTM model, not all core services
9. PDF analysis report with fixed template only, but the template cannot be easily modified by user or manager.
10. User operations history is stored in DB and displayed as a grid view on the website.

In addition, we have implemented an extra model for time-series prediction on our website. It is a statistical model (Holt-Winters) with a faster inferencing speed compared to LSTM. Together with the LSTM model, they can yield a more robust performance for both short-term and long-term predictions.

4.4 System Strength & Weakness

System strengths mainly involve the following:

- Native Python support where each commonly used python program can be encapsulated inside a view function in Django framework with little conversion issues.
- It is easy for users to use with clear guidance, and users do not need any professional training.

- The analysis result page provides some interactions to help users get more out of all different charts.
- Our website also allows users to download the analysis result in PDF format.

System weaknesses mainly involve the following:

Drawbacks involve relatively lower computational efficiency (Python implementation with unoptimised TensorFlow library), high coupling, low cohesion, little flexibility and customisation (We just provide one fixed chart for each core function, without considering users' preference. This could make our system less informative) as well as the fact that we could not provide more models to further reduce biases in our prediction which we would do differently. In addition, cookies and browser buffer-based data communication between frontend and backend is considered to be insecure, which could cause significant user privacy issues in actual application scenarios. We adapted popular implementations with some modifications in most algorithms instead of from scratch.

Overall performance of our team:

We were able to deliver project milestones on time with a certain level of quality based on the consensus reached after effective communication.

4.5 Evaluation for project highlight: LSTM prediction

Evaluation Strategy:

1. Calculate the time complexity.
2. Accuracy: Predict on a partition with ground truth and calculate the accuracy.

Conclusion:

This algorithm could genuinely predict future values while the error rate currently could not be reduced effectively.

Advantages:

1. Users can customise the forecast range.
2. The forecast result is shown right after historical data in the same graph to be more informative.

Disadvantages:

The forecast range should be limited to 30 days to avoid significant error.

Although the range can be customised, if the range is more than 30 days, in most cases, it tends to present an extreme error result (converge to a limit).

The cause can be traced back to the algorithm.

Firstly, the principle of the algorithm can be demonstrated as follows:

All "predictions" except the first one will be based on the immediate former prediction value. However, the current error rate cannot be reduced within a reasonable range in every iteration. Consequently, the error rate will accumulate. Thus, for large days of prediction, the ultimate error would be significant.

In addition, the method of error rate analysis in the Rabin Miller Primality Probability Test (Apdillah, Harahap and Khairina, 2017) is applicable. In our case:

k is denoted as an error rate factor indicating the error from this algorithm in each iteration. (if there is no error, k should be 1, k otherwise should be larger than 1)

T is a set $\{t_1, t_2, \dots, t_n\}$. Its elements represent the ground-truth value of the n^{th} iteration (no error).

A is a set $\{a_1, a_2, \dots, a_n\}$. Its elements represent the predicted value of the n^{th} iteration (indicate an error).

We then have relations as follows:

$t_1 = a_1$, we assume the first predicted value is accurate.

$a_n = k \times a_{n-1} - 1$ for $n \geq 2$

$t_n = 1 \times t_{n-1} - 1$ for $n \geq 2$

If there is no error, then

$$k = 1, a_n = k^{n-1}a_1 = 1^{n-1}a_1 = a_1$$

We also have $t_n = \dots = t_1 = a_n$

Thus $a_n = t_n$

Conclusion: If there is no error in this algorithm, the prediction will be accurate no matter the n .

If there exists an error, then $k > 1, a_n = k^{n-1}a_1$, while

$$t_n = \dots = t_1 = a_1$$

$$a_n = k^{n-1}a_1$$

Thus, the error coefficient will be $k^{n-1} - 1$. Since $k > 1$, if n is big enough, it could grow exponentially.

5 Future developments

In general, our system can be improved in terms of usability and flexibility.

Regarding user interface and experience, we will need to integrate some interactive guidance in each of our functions, which typically makes our website more user-friendly. When users choose a function, the website can guide the user step by step. In addition, for each function, there should be more customisable charts adaptable to user preferences. Finally, according to the analysis result, it is suggested that we can generate some specific advice to help users with their business decisions. This essentially requires more advanced techniques such as Natural Language Processing.

As for our system backend, we should add more novelty, robustness and efficiency into our algorithm implementation, especially for RFM and LSTM.

The DB design, implementation and integration could be more efficient and robust. The support of UI/UX flexibility and customisation for core functions should also be in place. Finally, we should have and are going to achieve higher cohesion & lower coupling in terms of programming to make the system more error-prone and future-proof.

6 Professional issues

Our group has carefully read the code of conduct issued by the British Computer Society (BCS, THE CHARTERED INSTITUTE FOR IT CODE OF CONDUCT FOR BCS MEMBERS, 2015) and followed it where applicable. In this section, we will critically demonstrate the extent of our compliance with the code of conduct.

6.1 Public Interest

In terms of public interest, we comply with the following principles. Our project employs a secure data pipeline to ensure the protection of user privacy and security. We use several libraries by third-party during our development process and respect their license. In addition, our project promotes equal access to our website since we aim to provide business intelligence for everyone and do not require professional training to use it.

6.2 Professional Competence and Integrity

Regarding professional competence and integrity, we comply with the following principles. We implemented three lightweight but useful functions in our website, which are all within our competence. In addition, professional integrity is strictly followed in our project. Despite some drawbacks to our project, we will accept criticism from our users and continuously improve our system according to user feedback.

6.3 Duty to Relevant Authority

Regarding the duty to the relevant authority, we comply with the following principles. We respect the requirements of our relevant authority and claim no conflict of interest. In addition, we have not authorised the disclosure of any confidential information for the benefit of a third party. Our source code will be released once the final portfolio is submitted, which ensures the transparency of our project.

6.4 Duty to the Profession

Regarding duty to the profession, we comply with the following principles. We managed to advocate the reputation and good standing of BCS. After our code release, we sincerely welcome fellow members to submit their pull requests and contribute to our project.

7 Bibliography

Apdillah, D.A., Harahap, M.K. and Khairina, N. (2017). Generating Mersenne Prime Number Using Rabin Miller Primality Probability Test to Get Big Prime Number in RSA Cryptography. *International Journal Of Information System & Technology*, [online] 1. doi:10.3.645.

BCS, THE CHARTERED INSTITUTE FOR IT CODE OF CONDUCT FOR BCS MEMBERS. (2015). [online] Available at: <https://www.bcs.org/media/2211/bcs-code-of-conduct.pdf> [Accessed 12 Apr. 2022].

CIPD (2020). *SWOT Analysis / Factsheets*. [online] CIPD. Available at: <https://www.cipd.co.uk/knowledge/strategy/organisational-development/swot-analysis-factsheet#gref> [Accessed 15 Apr. 2022].

Doyle, C. (2011). *A Dictionary of Marketing*. 4 ed ed. [online] Oxford University Press. Available at: <https://www.oxfordreference.com/view/10.1093/acref/9780199590230.001.0001/acref-9780199590230> [Accessed 2 Apr. 2022].

ECharts.apache.org. (n.d.). *Apache ECharts*. [online] Available at: <https://ECharts.apache.org> [Accessed 9 May 2022].

Schrepp, M., ThomaschewskiJ. and Hinderks, A. (2017). Design and Evaluation of a Short Version of the User Experience Questionnaire (UEQ-S). *International Journal of Interactive Multimedia and Artificial Intelligence*, [online] 4(Regular Issue). Available at: <https://www.ijimai.org/journal/bibcite/reference/2634> [Accessed 9 May 2022].

Test Documentation

Content

Overview.....	11
Testing Strategy.....	12
Unit Testing	12
Back-end Unit Testing	12
Front-end Unit Testing	19
Integration Testing	19
Interface Testing	19
Path Testing	22
System Testing	23
Functional Testing	23
Stress Testing.....	25
Compatibility Testing	26
Acceptance Testing	26
Alpha testing.....	26
Beta Testing	27
Summary	27
Testing Result Conclusion	27
Requirement Fulfilment Analysis.....	28

Overview

The test documentation involves a comprehensive description of the overall testing strategy of our Area38 AI-based Business Intelligence Website. It is divided into four sections, including unit testing, integration testing, system testing and acceptance testing. Additionally, there is a summary of testing results as well as the requirement fulfilment analysis at the end of the documentation. Established concurrently with the project design, this documentation effectively assists developers through the code implementation and testing process. It offers a possibility for the testing team to achieve maximum test coverage, thus enabling the high quality and outstanding performance of the Area38 AI-based Business Intelligence Website. Additionally, this testing documentation keeps track of developers' actions and whether they build the whole website based on the aims and objectives written in the requirements analysis. In this way, it ensures the functionalities presented by the website are consistent with customer requirements and do not deviate from the original intention. By testing the entire technical implementation and evaluating its usability and functionality dimensions, users can get a more reliable version of our website.

Testing Strategy

Unit Testing

Unit testing is considered the initial and most paramount phase in our whole testing process. Since our group developed frontend and backend separately in our implementation stage, we divided our testing process into frontend and backend unit testing, which can contribute to our timely discovery and correction of bugs and mistakes.

Backend Unit Testing

In the first stage of backend unit testing, the static testing method has been adopted by both backend and testing team to implement code inspection. Instead of running the program, errors are found by reading and evaluating every line of code after successfully implementing a method. Then, we used dynamic testing method to further observe potential bugs by writing test cases. Since our developers used MTV Django framework, an operation mechanism combining models in the backend implementation, our testing team created `test_models.py` and `test_views.py` to concentrate on testing database model and code logic processing. Here, we use a standard Python module which defines tests in the form of classes called `UnitTest` during our unit testing process.

1. Database Model Unit Testing

For our database Model testing which is related to the files of `model.py`, we used `django.test` framework which inherits Python's `unittest.TestCase` for testing three database models that have been created in the `model.py` file. When we intended to write test cases, our testing team used white boxing testing method to ensure all independent units in a module are tested at least once, thus enabling the comprehensiveness of the test case coverage.

1.1 Test Cases and Data

Our primary testing content for database models is to check whether the email address and its corresponding password have been stored correctly in a specific database. Since we have three databases named `UserInfo`, `UserFile` and `UserLog` that respectively store user information, user uploading file information and user login information, we designed three or four different test cases for each of them. The total amount of test cases we had run in this part is 30, and we finally classified them into nine tests. For `UserInfo` database, we made up three different assumptions to test whether the database has correctly stored the user's email address with its corresponding password. For `UserFile` database, we made up three different assumptions to test whether the database has correctly stored the user's email address with its corresponding file name and file description. For `UserLog` database, we make up three different assumptions to test whether the database has correctly stored the user's email address with its corresponding file name, action description and report name. The following forms have listed all test cases as well as their expected and actual results.

Database Model Python File (model.py) Unit Testing					
Test Case Filename:	test_model.py		Date Created:	2022.04.10	
Tester:	Yongyin Yang		Test Phase:	unit testing	
Index Number	Testing content	Test case	Expected Result	Actual Result	Pass/Fail
1	Testing UserInfo Database Model: whether the UserInfo database have stored the correct email address	email='Yongyin.Yang19@student.xjtlu.edu.cn' password='123456789'	email='Yongyin.Yang19@student.xjtlu.edu.cn'	email='Yongyin.Yang19@student.xjtlu.edu.cn'	Pass
	Testing UserInfo Database Model: whether the UserInfo database have stored the correct password	email='Yongyin.Yang19@student.xjtlu.edu.cn' password='123456789'	password='123456789'	password='123456789'	Pass
	Testing UserInfo Database Model: whether the UserInfo database have stored the user password to the corresponding correct email address	email='Yongyin.Yang19@student.xjtlu.edu.cn'	password='123456789'	password='123456789'	Pass
2	Testing UserInfo Database Model: whether the UserInfo database have stored the correct email address	email='Qiupu.Shen18@student.xjtlu.edu.cn' password='123456780'	email='Yongyin.Yang18@student.xjtlu.edu.cn'	email='Yongyin.Yang18@student.xjtlu.edu.cn'	Pass
	Testing UserInfo Database Model: whether the UserInfo database have stored the correct password	email='Qiupu.Shen18@student.xjtlu.edu.cn' password='123456780'	password='123456780'	password='123456780'	Pass
	Testing UserInfo Database Model: whether the UserInfo database have stored the user password to the corresponding correct email address	email='Qiupu.Shen18@student.xjtlu.edu.cn'	password='123456780'	password='123456780'	Pass
3	Testing UserInfo Database Model: whether the UserInfo database have stored the correct email address	email='Lixing.Lu20@student.xjtlu.edu.cn' password='123456789'	email='Lixing.Lu20@student.xjtlu.edu.cn'	email='Lixing.Lu20@student.xjtlu.edu.cn'	Pass
	Testing UserInfo Database Model: whether the UserInfo database have stored the correct password	email='Lixing.Lu20@student.xjtlu.edu.cn' password='123456789'	password='123456789'	password='123456789'	Pass
	Testing UserInfo Database Model: whether the UserInfo database have stored the user password to the corresponding correct email address	email='Lixing.Lu20@student.xjtlu.edu.cn'	password='123456789'	password='123456789'	Pass

4	Testing UserFile Database Model: whether the Userfile database have stored the correct email address	email='Yongyin.Yang19@student.xjtlu.edu.cn' fileName='Dataset1' fileDescription='This is a dataset of time series prediction.'	email='Yongyin.Yang19@student.xjtlu.edu.cn'	email='Yongyin.Yang19@student.xjtlu.edu.cn'	Pass
	Testing UserFile Database Model: whether the Userfile database have stored the correct file name corresponding to the email address	email='Yongyin.Yang19@student.xjtlu.edu.cn' fileName='Dataset1' fileDescription='This is a dataset of time series prediction.'	fileName='Dataset1'	fileName='Dataset1'	Pass
	Testing UserFile Database Model: whether the Userfile database have stored the correct file description corresponding to the email address	email='Yongyin.Yang19@student.xjtlu.edu.cn' fileName='Dataset1' fileDescription='This is a dataset of time series prediction.'	fileDescription='This is a dataset of time series prediction.'	fileDescription='This is a dataset of time series prediction.'	Pass
5	Testing UserFile Database Model: whether the Userfile database have stored the correct email address	email='Qiupu.Shen18@student.xjtlu.edu.cn' fileName='Dataset2' fileDescription='This is a dataset of RFM.'	email='Qiupu.Shen18@student.xjtlu.edu.cn'	email='Qiupu.Shen18@student.xjtlu.edu.cn'	Pass
	Testing UserFile Database Model: whether the Userfile database have stored the correct file name corresponding to the email address	email='Qiupu.Shen18@student.xjtlu.edu.cn' fileName='Dataset2' fileDescription='This is a dataset of RFM.'	fileName='Dataset2'	fileName='Dataset2'	Pass
	Testing UserFile Database Model: whether the Userfile database have stored the correct file description corresponding to the email address	email='Qiupu.Shen18@student.xjtlu.edu.cn' fileName='Dataset2' fileDescription='This is a dataset of RFM.'	fileDescription='This is a dataset of RFM.'	fileDescription='This is a dataset of RFM.'	Pass
6	Testing UserFile Database Model: whether the Userfile database have stored the correct email address	email='Lixing.Lu20@student.xjtlu.edu.cn' fileName='Dataset3' fileDescription='This is a dataset of Demand Curve analysis.'	email='Lixing.Lu20@student.xjtlu.edu.cn'	email='Lixing.Lu20@student.xjtlu.edu.cn'	Pass
	Testing UserFile Database Model: whether the Userfile database have stored the correct file name corresponding to the email address	email='Lixing.Lu20@student.xjtlu.edu.cn' fileName='Dataset3' fileDescription='This is a dataset of Demand Curve analysis.'	fileName='Dataset3'	fileName='Dataset3'	Pass
	Testing UserFile Database Model: whether the Userfile database have stored the correct file description corresponding to the email address	email='Lixing.Lu20@student.xjtlu.edu.cn' fileName='Dataset3' fileDescription='This is a dataset of Demand Curve analysis.'	fileDescription='This is a dataset of Demand Curve analysis.'	fileDescription='This is a dataset of Demand Curve analysis.'	Pass

7	Testing UserLog Database Model: whether the UserLog database have stored the correct email address	email='Yongyin.Yang19@student.xjtlu.edu.cn' fileName='Login' actionDescription='Yongyin, you have logined successfully!' reportName='Login Action1'	email='Yongyin.Yang19@student.xjtlu.edu.cn'	email='Yongyin.Yang19@student.xjtlu.edu.cn'	Pass
	Testing UserLog Database Model: whether the UserLog database have stored the correct file name corresponding to the email address	email='Yongyin.Yang19@student.xjtlu.edu.cn' fileName='Login' actionDescription='Yongyin, you have logined successfully!' reportName='Login Action1'	fileName='Login'	fileName='Login'	Pass
	Testing UserLog Database Model: whether the UserLog database have stored the correct action description corresponding to the email address	email='Yongyin.Yang19@student.xjtlu.edu.cn' fileName='Login' actionDescription='Yongyin, you have logined successfully!' reportName='Login Action1'	actionDescription='Yongyin, you have logined successfully!'	actionDescription='Yongyin, you have logined successfully!'	
	Testing UserLog Database Model: whether the UserLog database have stored the correct report name corresponding to the email address	email='Yongyin.Yang19@student.xjtlu.edu.cn' fileName='Login' actionDescription='Yongyin, you have logined successfully!' reportName='Login Action1'	reportName='Login Action1'	reportName='Login Action1'	Pass
8	Testing UserLog Database Model: whether the UserLog database have stored the correct email address	email='Qiupu.Shen18@student.xjtlu.edu.cn' fileName='Login' actionDescription='Qiupu, you have logined successfully!' reportName='Login Action2'	email='Qiupu.Shen18@student.xjtlu.edu.cn'	email='Qiupu.Shen18@student.xjtlu.edu.cn'	Pass
	Testing UserLog Database Model: whether the UserLog database have stored the correct file name corresponding to the email address	email='Qiupu.Shen18@student.xjtlu.edu.cn' fileName='Login' actionDescription='Qiupu, you have logined successfully!' reportName='Login Action2'	fileName='Login'	fileName='Login'	Pass
	Testing UserLog Database Model: whether the UserLog database have stored the correct action description corresponding to the email address	email='Qiupu.Shen18@student.xjtlu.edu.cn' fileName='Login' actionDescription='Qiupu, you have logined successfully!' reportName='Login Action2'	actionDescription='Qiupu, you have logined successfully!'	actionDescription='Qiupu, you have logined successfully!'	Pass
	Testing UserLog Database Model: whether the UserLog database have stored the correct report name corresponding to the email address	email='Qiupu.Shen18@student.xjtlu.edu.cn' fileName='Login' actionDescription='Qiupu, you have logined successfully!' reportName='Login Action2'	reportName='Login Action2'	reportName='Login Action2'	Pass

9	Testing UserLog Database Model: whether the UserLog database have stored the correct email address	email='Lixing.Lu20@student.xjtlu.edu.cn' fileName='Login' actionDescription='Yixing, you have logined successfully!' reportName='Login Action3'	email='Lixing.Lu20@student.xjtlu.edu.cn'	email='Lixing.Lu20@student.xjtlu.edu.cn'	Pass
	Testing UserLog Database Model: whether the UserLog database have stored the correct file name corresponding to the email address	email='Lixing.Lu20@student.xjtlu.edu.cn' fileName='Login' actionDescription='Yixing, you have logined successfully!' reportName='Login Action3'	fileName='Login'	fileName='Login'	Pass
	Testing UserLog Database Model: whether the UserLog database have stored the correct action description corresponding to the email address	email='Lixing.Lu20@student.xjtlu.edu.cn' fileName='Login' actionDescription='Yixing, you have logined successfully!' reportName='Login Action3'	actionDescription='Yixing, you have logined successfully!'	actionDescription='Yixing, you have logined successfully!'	Pass
	Testing UserLog Database Model: whether the UserLog database have stored the correct report name corresponding to the email address	email='Lixing.Lu20@student.xjtlu.edu.cn' fileName='Login' actionDescription='Yixing, you have logined successfully!' reportName='Login Action3'	reportName='Login Action3'	reportName='Login Action3'	Pass

1.2 Results and Analysis

According to the above test cases, the following screenshot shows the result as well as the evidence that we have performed all test cases in the command line.

```
(proj38) C:\Users\yangy\Desktop\Group-38--\Area_38>python manage.py test
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
.....
-----
Ran 9 tests in 0.042s

OK
Destroying test database for alias 'default'...
```

On this basis, we conclude that our website has passed all test cases that were designed for database models, including UserInfo, UserFile and UserLog. Therefore, for UserInfo database, the email address and its corresponding password will be stored correctly once a user registers for our website. When it comes to UserFile database, the email address with its corresponding file name and description will be saved after the user uploads a file for the specific functionality onto our website. Additionally, the third database named UserLog can successfully manage the data, including email address, file name, action description and report name, during the login process.

2. Views File Unit Testing

For our views file unit testing, our entire testing process can be divided into two phases.

In the first place, we intended to test the login function in the views.py file to check whether it satisfies our original requirements. Subsequently, we verified our register function by utilising different boundary values to see whether its implementation has met our standards.

2.1 Login function

When it comes to designing the test cases of the login function, we used equivalence partitioning method in black-box testing to classify those exhaustive test procedures into valid equivalence class and invalid equivalence class, thus enabling the integrity and representativeness of the designed test cases. For the valid equivalence class, we assumed the email address and password that users entered in are both correct. For the invalid equivalence class, we summarised the failure of login into the following conditions.

1. correct email address but incorrect password
2. correct password but incorrect email address
3. both password and email address are incorrect
4. both password and email address are null
5. password is null with correct email address
6. email address is null but password is correct
7. email address has already been registered

Based on the above conditions, we created test cases on each of them to test whether those cases will lead to login failures on our website which meets our design requirements.

2.2 Register function

For the register function in our views file, we used boundary value method in black-box testing to design our test cases. With the assistance of boundary value testing, we can ensure our register function behaves well even under some boundary conditions. Based on the original requirements that the user password length should be in the closed interval between 8 and 20, we set the password length to 7,8,9,19,20 and 21, respectively, for our test cases to check whether these passwords can pass the registration. Additionally, we also assumed some email addresses with the wrong format to test whether our website could prevent these email addresses from registering.

2.3 Test Cases and Data

The following form has listed all test cases about the login function and register function as well as their expected and actual results.

Views Python File (views.py) Unit Testing				
Test Case Filename:	test_views.py	Date Created:	2022.04.19	
Tester:	Yongyin Yang	Test Phase:	unit testing	
Index Number	Testing content	Expected Result	Actual Result	Pass/Fail
1	Login Function: correct email address but wrong password. 'email': 'Yongyin.Yang19@student.xjtlu.edu.cn', 'password': '12345678'	Login Failed	Login Failed	Pass
2	Login Function: correct password but email address is wrong 'email': 'Yongyin.Yang18@student.xjtlu.edu.cn', 'password': '123456789'	Login Failed	Login Failed	Pass
3	Login Function: both password and email address are wrong 'email': 'Yongyin.Yang18@student.xjtlu.edu.cn', 'password': '12345678'	Login Failed	Login Failed	Pass
4	Login Function: both password and email address are null 'email': '', 'password': ''	Login Failed	Login Failed	Pass
5	Login Function: password is null but email address is correct 'email': 'Yongyin.Yang19@student.xjtlu.edu.cn', 'password': ''	Login Failed	Login Failed	Pass
6	Login Function: email address is null but password is correct 'email': '', 'password': '123456789'	Login Failed	Login Failed	Pass
7	Login Function: both email address and password are correct 'email': 'Yongyin.Yang19@student.xjtlu.edu.cn', 'password': '123456789'	Login Successfully status_code=302	Login Successfully status_code=302	Pass
8	Register Function: email address has already been registered 'email': 'Yongyin.Yang19@student.xjtlu.edu.cn', 'password': '123456789'	Login Failed	Login Failed	Pass
9	Register Function: the length of the password is 7 'email': 'Yongyin.Yang19@student.xjtlu.edu.cn', 'password': '1234567'	Register Failed	Register Failed	Pass
10	Register Function: the length of the password is 8 'email': 'Yongyin.Yang19@student.xjtlu.edu.cn', 'password': '123456789'	Register Successfully status_code=302	Register Successfully status_code=302	Pass
11	Register Function: the length of the password is 9 'email': 'Yongyin.Yang19@student.xjtlu.edu.cn', 'password': '123456789'	Register Successfully status_code=302	Register Successfully status_code=302	Pass
12	Register Function: the length of the password is 19 'email': 'Yongyin.Yang19@student.xjtlu.edu.cn', 'password': '123457891234567891'	Register Successfully status_code=302	Register Successfully status_code=302	Pass
13	Register Function: the length of the password is 20 'email': 'Yongyin.Yang19@student.xjtlu.edu.cn', 'password': '1234578912345678910'	Register Successfully status_code=302	Register Successfully status_code=302	Pass
14	Register Function: the length of the password is 21 'email': 'Yongyin.Yang19@student.xjtlu.edu.cn', 'password': '1123457891234567891'	Register Failed	Register Failed	Pass
15	Register Function: the email with wrong format case1 'email': 'Yongyin.Yang19@', 'password': '123456789'	Register Failed	Register Failed	Pass
16	Register Function: the email with wrong format case2 'email': 'Yongyin.Yang ', 'password': '123456789'	Register Failed	Register Failed	Pass

2.4 Results and Analysis

According to the above test cases, the following screenshot shows the result as well as the evidence that we have performed all test cases in the command line.

```

userName: Yongyin18
email: Yongyin.Yang18@student.xjtlu.edu.cn
passwd: 1234567890123456789101
.we post!
userName: Yongyin19
email: Yongyin.Yang19@student.xjtlu.edu.cn
passwd: 12345678
register failed, user already exists
.
-----
Ran 13 tests in 0.056s

OK
Destroying test database for alias 'default'...
```

Based on the above result, as well as the hypothetical data designed for the register and login function, we can draw a conclusion that our website has passed all test cases. Therefore, for register functionality, it satisfies our original requirement that the email address should not have been registered if the user intends to use this email for registration. Additionally, the email for registering should follow a specific format, and its password should be in the closed interval between 8 and 20. For login functionality,

it meets the requirement that only when both email address and password are not null and current can ensure a successful login process.

Frontend Unit Testing

However, due to the time limitation and current limited knowledge we owned for software testing, our testing team only adopted static testing method for our frontend unit testing.

Integration Testing

After passing all aspects of unit testing, our testing team put emphasis on integration testing which involves interface testing and path testing. Both of them were implemented by bottom-up integration testing method.

Interface Testing

For the interface testing, we focused on the interface connection and data flow transmission between modules. In the first step, we made up our decision to use bottom-up integration testing strategy which aims at implementing low-level modules test firstly and then further use them to facilitate testing of higher-level modules. In other words, once the lower-level modules are tested and integrated by our testers, the next level modules will be formed. In this way, it facilitates our testing team to locate specific faults in a much faster way. Subsequently, we prepared our test cases and decided the priority of each interface for testing.

The following are the sequence and content of our interface testing.

1. function response code testing
2. template rendering testing
3. POST Interface Testing

1. Function Response Code Testing

When it comes to testing each function response code, we intended to test whether the linked page exists. If the linked page does not exist, the function response code will fail which is not equal to 200.

1.1 Test Cases and Data

The following forms have listed all test cases as well as their expected and actual results.

Views Python File (views.py) Unit Testing				
Test Case Filename:	test_views.py	Date Created:	2022.04.15	
Tester:	Yongyin Yang	Test Phase:	integration testing	
Index Number	Testing content	Expected Result	Actual Result	Pass/Fail
1	Test login function response code	status_code=200	status_code=200	Pass
2	Test register function response code	status_code=200	status_code=200	Pass
3	Test forget function response code	status_code=200	status_code=200	Pass
4	Test SDChart function response code	status_code=200	status_code=200	Pass
5	Test RFMChart function response code	status_code=200	status_code=200	Pass
6	Test TSChart function response code	status_code=200	status_code=200	Pass
7	Test uploadfile function response code	status_code=200	status_code=200	Pass
8	Test home function response code	status_code=200	status_code=200	Pass
9	Test demend function response code	status_code=200	status_code=200	Pass
10	Test rfm function response code	status_code=200	status_code=200	Pass
11	Test xts function response code	status_code=200	TypeError	Fail
12	Test logout function response code	status_code=200	status_code=200	Pass
13	Test hw function response code	status_code=200	status_code=200	Pass
14	Test AboutUs function response code	status_code=200	status_code=200	Pass
15	Test guidance function response code	status_code=200	status_code=200	Pass
16	Test Admin function response code	status_code=200	status_code=200	Pass
17	Test pdf_download function response code	status_code=200	status_code=200	Pass
18	Test history function response code	status_code=200	status_code=200	Pass
19	Test history/api function response code	status_code=200	status_code=200	Pass

1.2 Results and Analysis

According to the above test cases, the following screenshot shows the result as well as the evidence that we have performed all test cases in the command line.

```

Traceback (most recent call last):
  File "C:\Users\yangy\Desktop\Group-38--\Area_38\Area_38_app\test_views.py", line 75, in test_xts_GET1
    response= self.client.get(self.xts_url)
  File "C:\ProgramData\Anaconda3\envs\proj38\lib\site-packages\django\test\client.py", line 742, in get
    response = super().get(path, data=data, secure=secure, **extra)
  File "C:\ProgramData\Anaconda3\envs\proj38\lib\site-packages\django\test\client.py", line 398, in get
    **extra,
  File "C:\ProgramData\Anaconda3\envs\proj38\lib\site-packages\django\test\client.py", line 473, in generic
    return self.request(**r)
  File "C:\ProgramData\Anaconda3\envs\proj38\lib\site-packages\django\test\client.py", line 719, in request
    self.check_exception(response)
  File "C:\ProgramData\Anaconda3\envs\proj38\lib\site-packages\django\test\client.py", line 580, in check_exception
    raise exc_value
  File "C:\ProgramData\Anaconda3\envs\proj38\lib\site-packages\django\core\handlers\exception.py", line 47, in inner
    response = get_response(request)
  File "C:\ProgramData\Anaconda3\envs\proj38\lib\site-packages\django\core\handlers\base.py", line 181, in _get_response
    response = wrapped_callback(request, *callback_args, **callback_kwargs)
  File "C:\Users\yangy\Desktop\Group-38--\Area_38\Area_38_app\views.py", line 643, in xts
    uploadfilepath = "/" + user_email + "/upload/"
TypeError: can only concatenate str (not "NoneType") to str

```

```

Ran 14 tests in 4.960s
FAILED (errors=1)
Destroying test database for alias 'default'...
(proj38) C:\Users\yangy\Desktop\Group-38--\Area_38>

```

When our testing team attempted to receive the response code from the xts function, the compiler returned a type error that demonstrated that there is no permission to concatenate other types of variables other than type 'str' when assigning the variable uploadfilepath. Since the actual result was different from the result that we expected, this test failed in our initial integration testing process. Afterwards, our testing team immediately gave an account of this problem to backend developers.

2. Template Rendering Testing

When it comes to template rendering testing, it aims at checking which template response is used to render the specified function. During our test, we intended to test

whether the expected render template is equal to the actual result.

2.1 Test Cases and Data

The following forms have listed all test cases as well as their expected and actual results.

Views Python File (views.py) Unit Testing				
Test Case Filename:	test_views.py	Date Created:	2022.04.17	
Tester:	Yongyin Yang	Test Phase:	unit testing	
Index Number	Testing content	Expected Result	Actual Result	Pass/Fail
1	check which template login response is rendered with	response='index.html'	'index.html'	Pass
2	check which template register response is rendered with	response='index.html'	'index.html'	Pass
3	check which template forget response is rendered with	response='index.html'	'index.html'	Pass
4	check which template SDChart response is rendered with	response='index.html'	'index.html'	Pass
5	check which template RFMChart response is rendered with	response='index.html'	'index.html'	Pass
6	check which template TSChart response is rendered with	response='index.html'	'index.html'	Pass
7	check which template uploadfile response is rendered with	response='index.html'	'index.html'	Pass
8	check which template home response is rendered with	response='index.html'	'index.html'	Pass
9	check which template history response is rendered with	response='index.html'	ValueError	Fail
10	check which template AboutUs response is rendered with	response='index.html'	'index.html'	Pass
11	check which template guidance response is rendered with	response='index.html'	'index.html'	Pass

2.2 Results and Analysis

According to the above test cases, the following screenshot shows the result as well as the evidence that we have performed all test cases in the command line.

```

self.check_response(response, callback)
File "C:\ProgramData\Anaconda3\envs\proj38\lib\site-packages\django\core\handlers\base.py", line 311, in check_response
    "instead." % name
ValueError: The view Area_38_app.views.history didn't return an HttpResponse object. It returned None instead.

-----
Ran 12 tests in 0.080s

FAILED (errors=1)
Destroying test database for alias 'default'...
```

When our testing team attempted to receive the response from the history function to check which template the history response is rendered with, the compiler returned a value error which demonstrated that the history function returned 'None' instead of an HttpResponse object. Since the actual result was different from the result that we expected, this test failed in our integration testing process. Afterwards, our testing team immediately gave an account of this problem to backend developers for error correction.

3. POST Interface Testing

When it comes to POST interface testing, it aims at checking whether our website is capable of submitting data to the server for processing. During our test, we used equivalence partitioning method in black-box testing to test whether functions including login, register, uploadfile and admin can work with both data transmission and no data input.

3.1 Test Cases and Data

The following forms have listed all test cases as well as their expected and actual results.

Views Python File (views.py) Unit Testing				
Test Case Filename:	test_views.py	Date Created:	2022.04.17	
Tester:	Yongyin Yang	Test Phase:	unit testing	
Index Number	Testing content	Expected Result	Actual Result	Pass/Fail
1	check POST interface of login function with no data	status_code=200	status_code=200	Pass
2	check POST interface of login function with data. UserInfo.objects.first().email='Yongyin.Yang19@student.xjtlu.edu.cn'	TRUE	TRUE	Pass
3	check POST interface of register function with no data	status_code=200	status_code=200	Pass
4	check POST interface of register function with data. UserInfo.objects.first().email='Yongyin.Yang19@student.xjtlu.edu.cn'	TRUE	TRUE	Pass
5	check POST interface of uploadfile function with no data	status_code=200	status_code=200	Pass
6	check POST interface of Admin function with no data	status_code=200	status_code=200	Pass

3.2 Results and Analysis

According to the above test cases, the following screenshot shows the result as well as the evidence that we have performed all test cases in the command line.

```
(proj38) C:\Users\yangy\Desktop\Group-38--\Area_38>python manage.py test
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
.....no file
.
-----
Ran 6 tests in 0.031s
OK
Destroying test database for alias 'default'...
```

Based on the result listed above, we can draw a conclusion that our website has passed all test cases in POST interface testing.

Path Testing

1. URL Testing

URL Testing is another module in integration testing. Here, we are going to test whether all links link to the correct page as indicated. Since all URLs had been written in one file called urls.py, our testing team created a test_urls.py for URL connection testing.

1.1 Test Cases and Data

The following forms have listed all test cases as well as their expected and actual results.

Urls Python File (urls.py) Unit Testing				
Test Case Filename:	test_urls.py	Date Created:	2022.04.22	
Tester:	Yongyin Yang	Test Phase:	integration testing	
Index Number	Testing content	Expected Result	Actual Result	Pass/Fail
1	test whether the login link displays properly	TRUE	TRUE	Pass
2	test whether the register link displays properly	TRUE	TRUE	Pass
3	test whether the forget link displays properly	TRUE	TRUE	Pass
4	test whether the SDChart link displays properly	TRUE	TRUE	Pass
5	test whether the RFMChart link displays properly	TRUE	TRUE	Pass
6	test whether the TSChart link displays properly	TRUE	TRUE	Pass
7	test whether the uploadfile link displays properly	TRUE	TRUE	Pass
8	test whether the home link displays properly	TRUE	TRUE	Pass
9	test whether the demand link displays properly	TRUE	TRUE	Pass
10	test whether the rfm link displays properly	TRUE	TRUE	Pass
11	test whether the xts link displays properly	TRUE	TRUE	Pass
12	test whether the logout link displays properly	TRUE	TRUE	Pass
13	test whether the holt winter link displays properly	TRUE	TRUE	Pass
14	test whether the about us link displays properly	TRUE	TRUE	Pass
15	test whether the guidance link displays properly	TRUE	TRUE	Pass
16	test whether the Admin link displays properly	TRUE	TRUE	Pass
17	test whether the pdf download link displays properly	TRUE	TRUE	Pass
18	test whether the history link displays properly	TRUE	TRUE	Pass
19	test whether the history API link displays properly	TRUE	TRUE	Pass

1.2 Results and Analysis

According to the above test cases, the following screenshot shows the result as well as the evidence that we have performed all test cases in the command line.

```
(proj38) C:\Users\yangy\Desktop\Group-38--\Area_38>python manage.py test
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
.....
-----
Ran 19 tests in 0.022s

OK
Destroying test database for alias 'default'...
```

Based on the result listed above, we can draw a conclusion that our website has passed all test cases in URL Testing. Therefore, all links link to the correct page as indicated, which enables our website to run smoothly.

System Testing

After accomplishing both unit testing and integration testing, our testing team focused on the system testing which inspected whether there were faults in our design and implementation process. In this phase, we divided our testing process into three parts which are functional testing, stress testing and compatibility testing. The following are their respective detailed descriptions.

Functional Testing

1. Basic Functionality

Among all kinds of system testing, we give priority to functional testing since it acts as the most significant measurable indicator of whether our website is performing in accordance with pre-determined demands. During the functional testing, we initially implemented the smoke testing to quickly have verification whether the basic functionality of the website is defective.

1.1 Test Cases and Data

All following test cases are based on the aims and objectives written in the requirement analysis.

Website Functional Testing		
Tester:	Yongyin Yang	Test Phase: System Testing
Index Number	Test Cases	Implemented/ Not Implemented
1	Implement user account operations (register/login/change password/delete account data) for website service	Implemented
2	Implement a real-time interactive demand curve to provide optimized pricing strategy for a higher profit or customized goal	Implemented
3	Implement algorithms such as Recency, Frequency, Monetary Value (RFM) analysis to find optimal/potential customer group	Implemented
4	Implement a business metrics forecast dashboard derived from Statistical and Machine Learning (ML) models	Implemented
5	Enable users to upload required files. If there is an upload failure like network or browser issue, remind users to upload again and clear the incomplete buffer data.	Implemented
6	Enable users to download detailed report about the corresponding service	Implemented
7	Enable administrator to monitor database user accounts and transactions	Implemented
8	Enable administrator to upgrade BI services, such as Statistical/ML models	Implemented
9	Enable Product Manager to modify contents in report templates	Implemented
10	Enable database to store analysis history of a certain account for user to reference	Implemented

1.2 Results and Analysis

From the table, we can observe that our website has implemented all basic functionalities without obvious defects. Therefore, we can draw a conclusion that our website is generally consistent with our objectives in the requirement analysis.

2. Extra Functionality

Additionally, we used black-box testing method to check some subtle functionalities of our website without understanding its internal structure and processing procedure. The following chart shows all test cases and their corresponding executive results.

2.1 Test Cases and Data

Website Functional Testing		
Tester:	Yongyin Yang	Test Phase: System Testing
Index Number	Test Cases	Implemented/ Not Implemented
1	Implement an accordion menu on the left of our website	Implemented
2	Implement the search box to get specific information in the User Guidance page	Implemented
3	Enable users to log out the website at any time when they intend to leave	Implemented
4	Enable users to skip the registration process if they have no interest in it	Implemented
5	Implement the Boxplot which visualises the statistical distribution of the raw input data such as range, mean, variance and interquartile range	Implemented
6	Enable users to zoom in and out by scrolling up and down when they are looking at the Target Customer Group Analysis Graph	Implemented
7	Enable users to see group name and total group amount when they place the pointer over each customer group block of the Target Customer Group Analysis Graph	Implemented
8	Enable users to upload any format of dataset files when they intend to use Demand Curve, RFM Prediction or Time Series Prediction	Not Implemented
9	Enable users to rechoose dataset files in the uploading process	Implemented
10	Enable users to change the start and end of the x-axis to zoom in or out the whole line graph in the time series prediction functionality	Implemented

2.2 Results and Analysis

From the above table, we can observe that our website has implemented 90% of the extra functionalities during our functional testing process. However, aiming to ensure a successful analysis of Demand Curve Functionality, RFM Prediction Functionality and Time Series Prediction, our website only allows users to upload files in the specified format. For both Demand Curve Functionality and RFM Prediction Functionality, the user is only allowed to upload files in the format of csv. When it comes to the Time Series Prediction Functionality, the user should ensure the format of their dataset files that the first column of the dataset must be "Time". Therefore, our website cannot satisfy users to upload any format of dataset files in the current stage.

Stress Testing

The main purpose of stress testing for our website is to test the maximum number of users that the server can support beyond normal operational capacity of a dramatic increase in traffic. For stress testing, we used manual testing strategy to let six users login to our website concurrently and then subsequently checked our website that can still run normally and smoothly. The final testing result indicated that our website has the carrying capacity of at least six users to perform operations such as registering, uploading files etc., simultaneously on our website.

Compatibility Testing

The significance of compatibility testing is to ensure whether our website works under different configurations. With the assistance of the automatic tool called PowerMapper, we checked the compatibility of our website with different browsers like Firefox, Google Chrome, Internet Explorer etc. The following shows the executive results of our website.

1.1 Results and Analysis

Browser Compatibility	
Android Browser Compatibility	✓ Checked latest version
Chrome Browser Compatibility	✓ Checked latest version
Edge Browser Compatibility	✓ Checked latest version
Internet Explorer Browser Compatibility	✓ Checked latest version
Firefox Browser Compatibility	✓ Checked latest version
Opera Browser Compatibility	✓ Checked latest version
Safari Browser Compatibility	✓ Checked latest version
iPhone/iPad Browser Compatibility	✓ Checked from version 13

From the table, we can observe that our website has passed all tests under the listed hypothetical scenarios. Therefore, we can draw a conclusion that our website works well under different browsers.

Acceptance Testing

Acceptance testing is the final stage among all our four testing phases. Aiming to guarantee the practicability of our website for end-users, we used two major methods involving alpha testing and beta testing for our whole acceptance testing process. Firstly, we implemented alpha testing to validate the matching degree between our website and requirement analysis. For the next step, we conducted beta testing by using the questionnaire to carry out direct feedback from real users. The following are their respective detailed descriptions.

Alpha testing

For the Alpha testing, it is our initial phase of acceptance testing to validate whether a new product will perform as the requirement analysis expected. It is carried out by our testers and developers to identify and correct all possible issues and potential bugs before releasing the software to beta testers. Here, we simulate real users by using both black box testing and white box testing techniques to test our website functionality and usability.

Beta Testing

After conducting the alpha testing, we implemented beta testing which is considered as a form of external user acceptance testing method to carry out feedback from real users in a real environment. Since we can receive direct suggestions from users, we can gain a profound understanding of whether our website can cater to the requirements of different commercial entities and individuals to a great extent. For our implementation of beta testing, we invited ten participants (including one female) to complete the User Experience Questionnaire Short Version after using our website. The following is the template of our questionnaire.

obstructive	o o o o o o o	supportive
complicated	o o o o o o o	easy
inefficient	o o o o o o o	efficient
confusing	o o o o o o o	clear
boring	o o o o o o o	exciting
not interesting	o o o o o o o	interesting
conventional	o o o o o o o	inventive
usual	o o o o o o o	leading edge

Based on the feedback we received from all questionnaires, we obtained the mean overall score of 1.89 (s.d = .80), mean Pragmatic Quality Score of 2.10 (s.d. = .75), and mean Hedonic Quality score of 1.68 (s.d. = .92), where scores' range is (-3 ~ 3). All our scores achieve the excellent benchmark of UEQ-S. In this way, it can, to some extent, validate that our website could generally meet the aim and objectives written in the requirement analysis, especially in terms of user-friendliness and clarity.

Summary

Testing Result Conclusion

In conclusion, the above results show that the total number of test cases we have designed is 80. For our Back-end unit testing, all test cases have passed the tests, while when it comes to integration testing, there are two errors in the history function and xts function, respectively, during the backend development. When our testing team attempted to receive the response from the history function to check which template history response was rendered with, the compiler returned a value error which demonstrated that the history function returned 'None' instead of an HttpResponse object. Additionally, the compiler returned a type error, which means there is no permission to concatenate other types of variables other than type 'str' when we attempted to receive the response code from the xts function. Once our testing team had inspected these bugs, we immediately gave an account of these

problems to backend developers for further error correction.

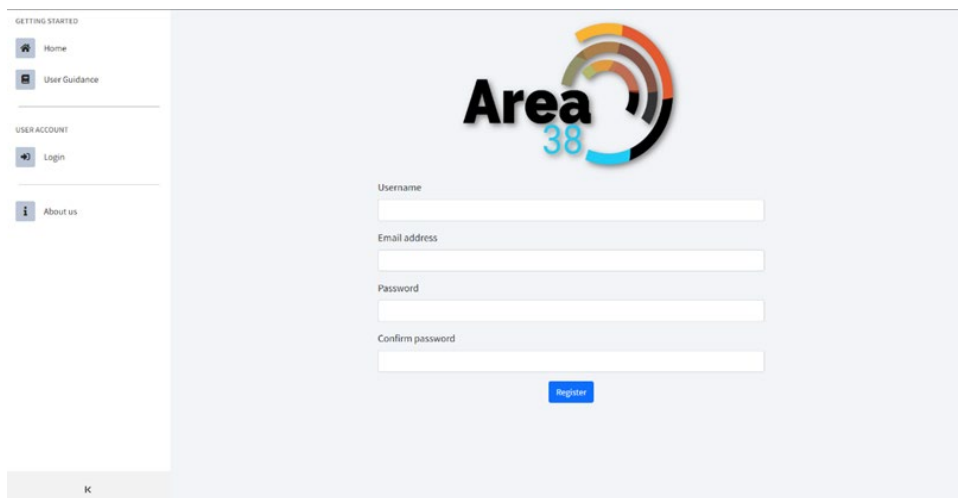
In the next stage, during the phase of system testing, although our website only allows users to upload files in a specified format which is not consistent with our original requirement that users can upload any type of file, it has implemented all basic functionalities without obvious defects in functional testing. Moreover, our website did an outstanding performance in implementing stress testing and compatibility testing. Finally, in the acceptance testing, we conducted both alpha testing and beta testing to carry out feedback from real users in a real environment. The result from beta testing illustrated that all our scores collected from the questionnaire had achieved the excellent benchmark of UEQ-S. In this way, it can further validate that our website is consistent with the aim and objectives written in the requirement analysis.

Requirement Fulfilment Analysis

According to the requirement analysis, we have satisfied all requirements that were written in our aim and objective. The implemented requirements are listed with evidence in the following.

1. Implement user account operations (register/login/change password/logout) for website service.

For the register function,



GETTING STARTED

- Home
- User Guidance

USER ACCOUNT

- Login

About us

Area 38

Username

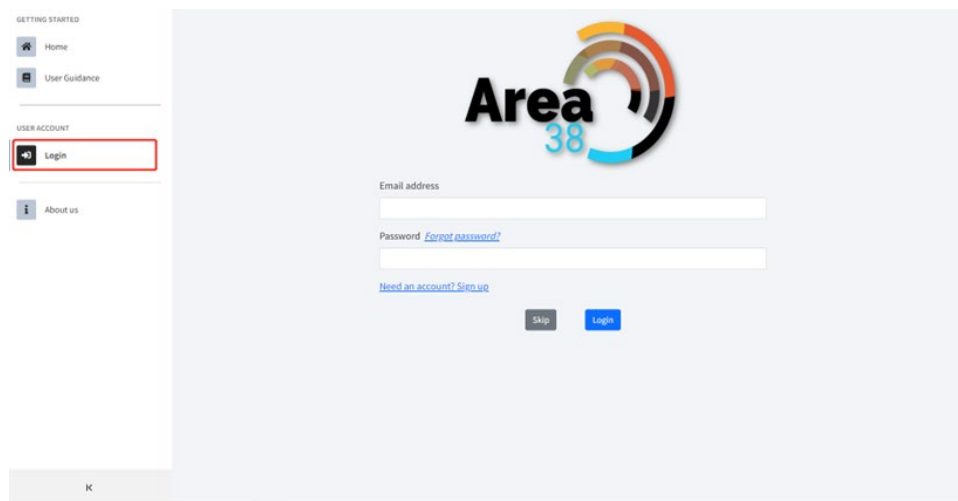
Email address

Password

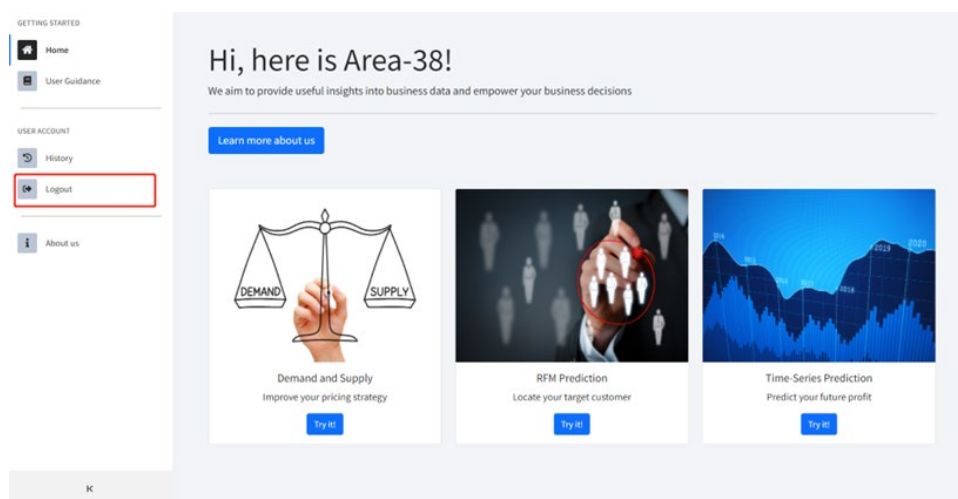
Confirm password

Register

For the login function,



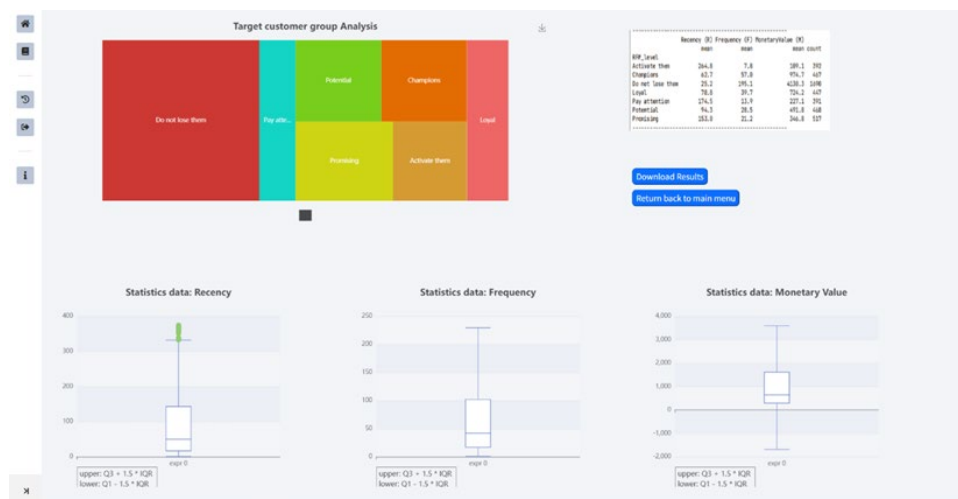
For the logout function,



2. Implement a real-time interactive demand curve to provide an optimised pricing strategy for a higher profit or customised goal.



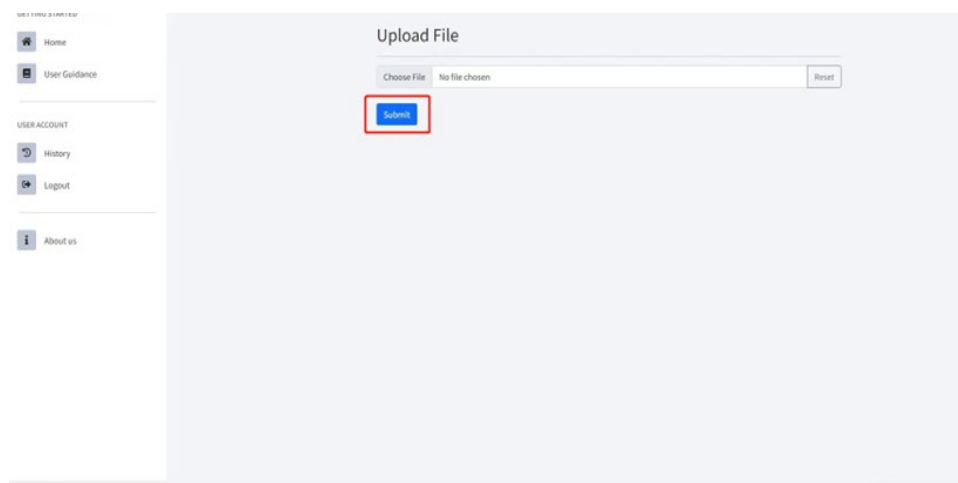
- Implement algorithms such as Recency, Frequency, Monetary Value (RFM) analysis to find optimal/potential customer groups.



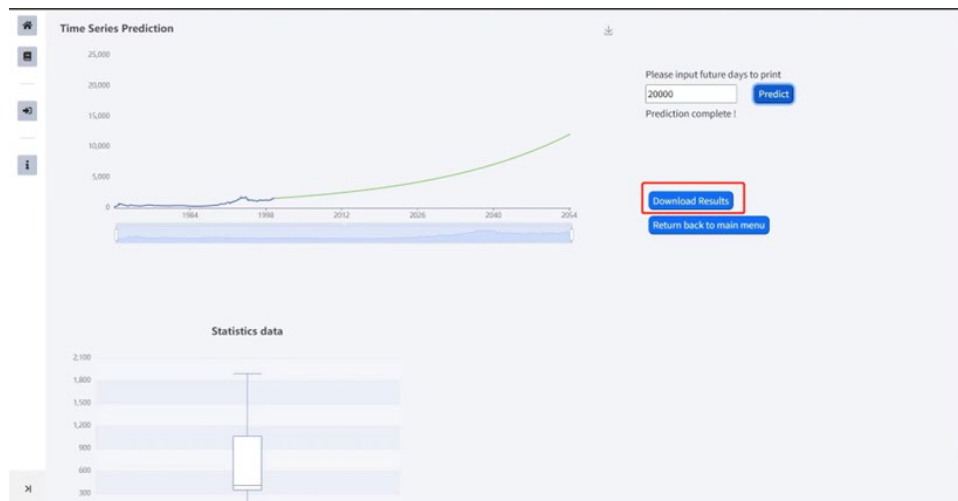
- Implement a business metrics forecast dashboard derived from Statistical/Machine Learning (ML) models.



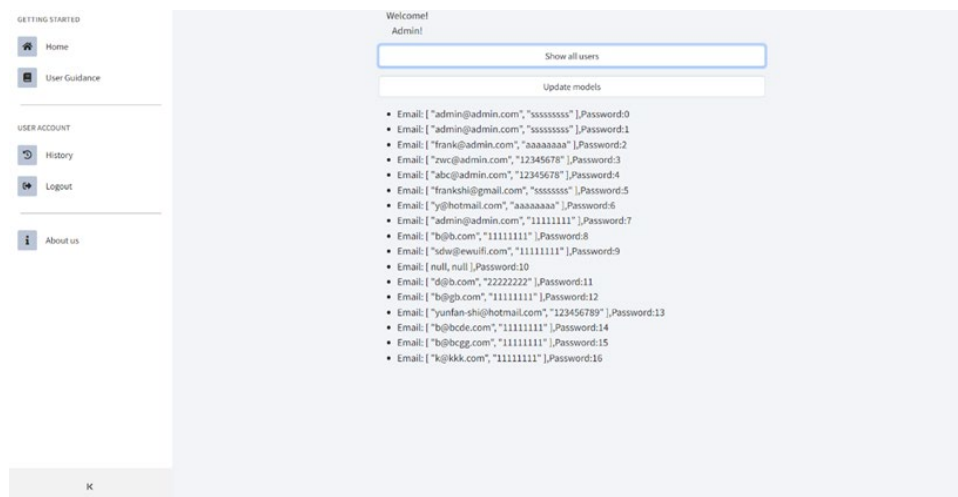
5. Enable users to upload required files. If there is an upload failure like a network or browser issue, remind users to upload again and clear the incomplete buffer data.



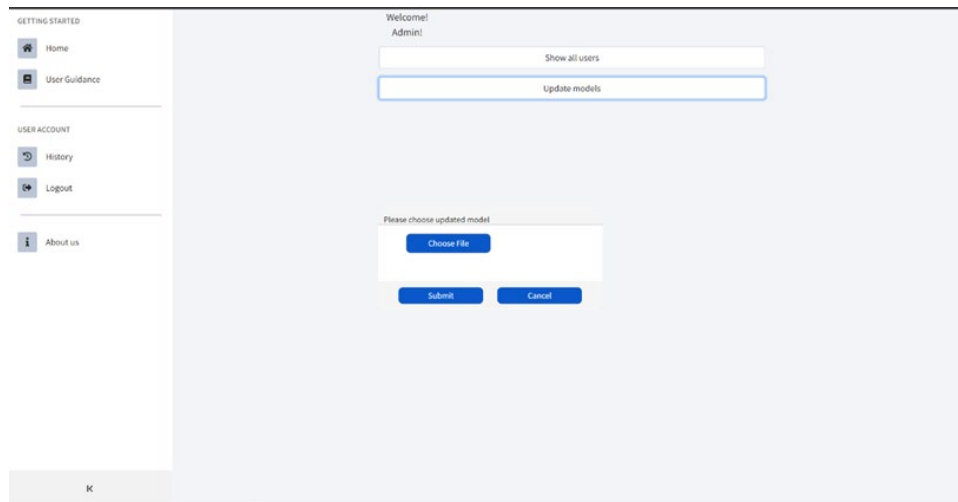
6. Enable users to download detailed reports about the corresponding service



7. Enable administrator to monitor database user accounts and transactions.



8. Enable service engineers to upgrade BI services, such as Statistical/ML models.

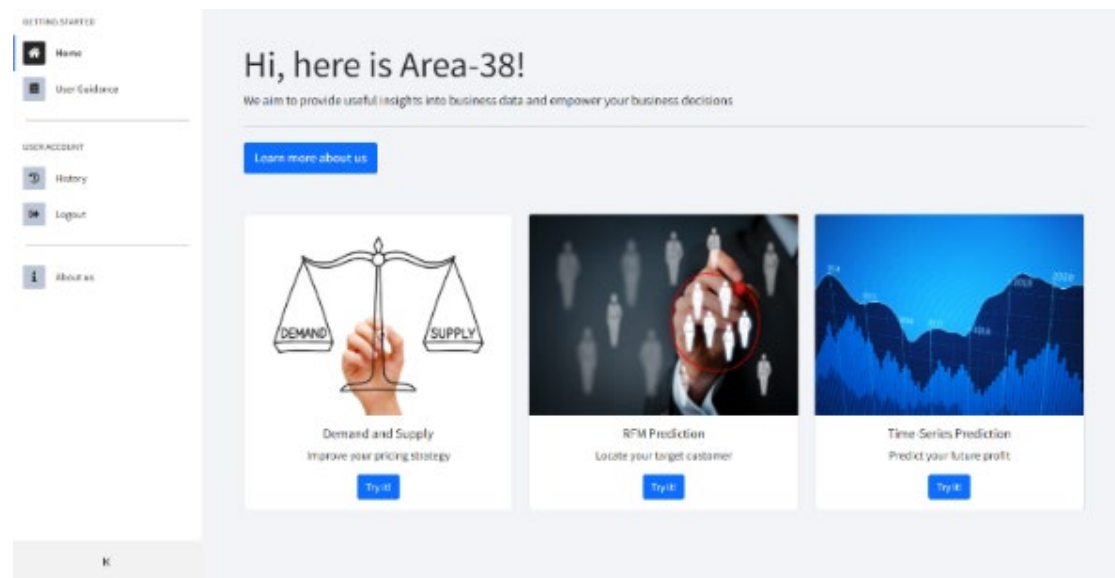


9. Enable database to store analysis history of a certain account for user to reference.

The screenshot shows a web application interface with a sidebar on the left. The sidebar has a 'GETTING STARTED' section with 'Home' and 'User Guidance', and a 'USER ACCOUNT' section with 'History', 'Logout', and 'About us'. The 'History' link is highlighted with a red box. The main content area displays a table titled 'History Actions'.

Date	Uploaded File	Action
2022-04-18 22:23:28.439279+00:00	demand.csv	1
2022-04-18 22:25:11.803629+00:00	demand.csv	1
2022-04-18 22:26:34.520320+00:00	demand.csv	1
2022-04-18 22:26:59.786567+00:00	demand.csv	1
2022-04-18 22:30:01.734567+00:00	demand.csv	1
2022-04-18 22:30:56.970214+00:00	demand.csv	1
2022-04-18 22:31:35.310938+00:00	demand.csv	1
2022-04-18 22:32:37.534789+00:00	demand.csv	1
2022-04-18 22:33:11.407829+00:00	demand.csv	1
2022-04-18 22:37:16.109422+00:00	rfm.csv	2
2022-04-18 22:42:46.292306+00:00	demand.csv	1

Sample Screenshots:



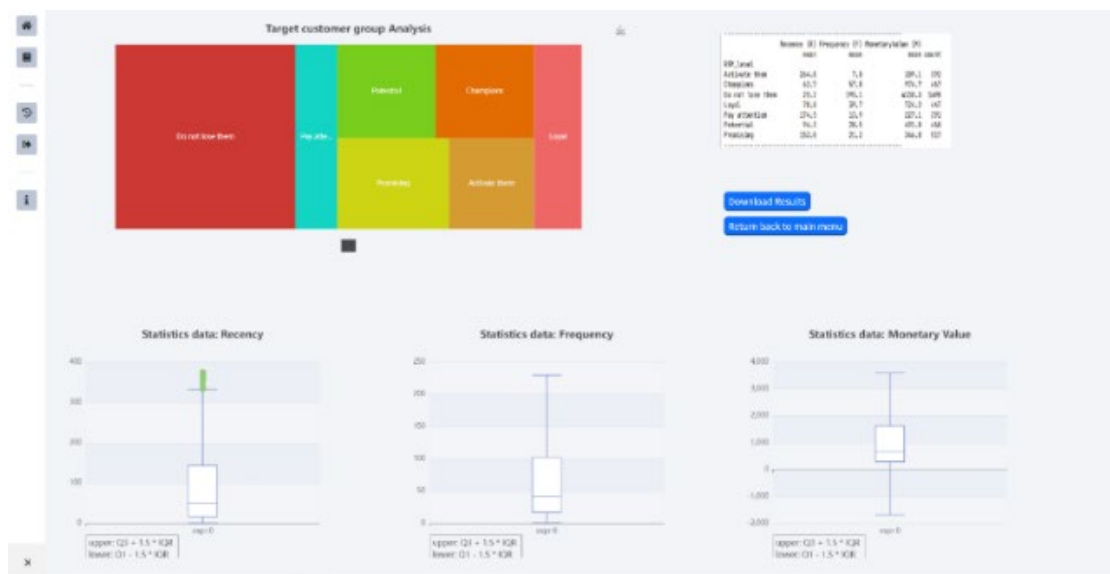
User Dashboard

History Actions		
Date	Uploaded File	Action
2022-05-03 21:12:21.906744+00:00	gold_price_data.csv	3.2
2022-04-27 09:24:57.728252+00:00	gold_price_data.csv	3.2
2022-04-27 09:13:34.378314+00:00	gold_price_data.csv	3.1
2022-04-24 22:38:19.090268+00:00	gold_price_data.csv	3.1
2022-04-21 22:10:00.55138100:00	rfm.csv	2
2022-04-21 13:37:05.068922+00:00	gold_price_data.csv	3.2
2022-04-21 13:36:29.377361+00:00	gold_price_data.csv	3.1
2022-04-21 13:35:26.516554+00:00	gold_price_data.csv	3.2
2022-04-21 13:24:46.832088+00:00	gold_price_data.csv	3.2
2022-04-21 13:24:16.305447+00:00	gold_price_data.csv	3.1
2022-04-21 13:23:33.967009+00:00	rfm.csv	2
2022-04-21 13:23:43.586370+00:00	demand.csv	1
2022-04-21 13:21:41.142192+00:00	gold_price_data.csv	3.1
2022-04-21 13:21:12.013189+00:00	gold_price_data.csv	3.2
2022-04-21 13:20:31.629501+00:00	rfm.csv	2

User analysis history



Demand Curve



RFM



LSTM



Holt-Winters