

A Smart Energy Management and Monitoring System

EBU5304 SOFTWARE ENGINEERING

Group 102

**Wenhao Zhang, Yang Ding, Shiyuan Wang,
Jindou Wei, Keke Wang, Zhe Wang**

Content

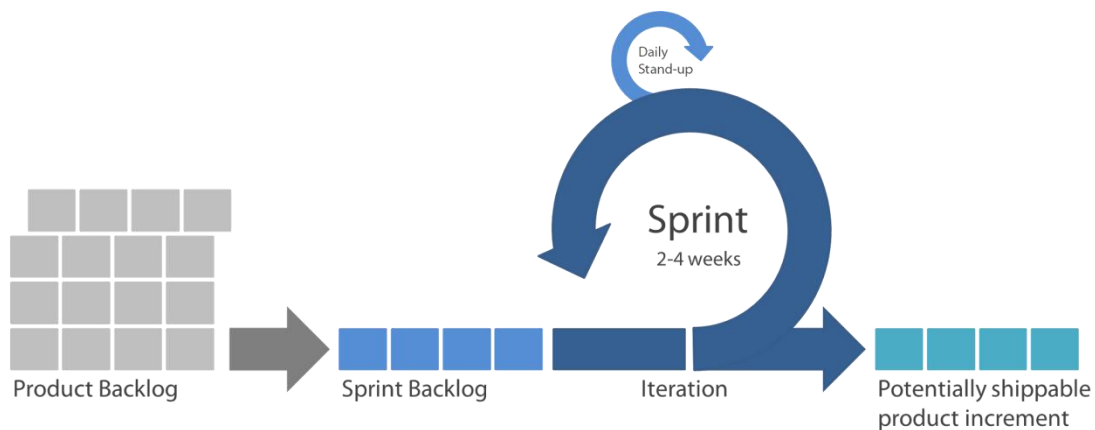
1. Project Management.....	1
1.1 Introduction.....	1
1.2 Agile project management.....	1
1.3 Human resource management.....	2
1.4 Risk management.....	2
1.4.1 Risk Analysis.....	2
1.4.2 Risk planning.....	2
1.5 Agile team working.....	3
1.5.1 Pair Programming.....	3
1.5.2 Meetings.....	3
1.6 Schedule and Milestones.....	3
2. Requirement.....	4
2.1 Fact-finding Techniques.....	4
2.1.1 Background reading.....	4
2.1.2 Observation.....	4
2.2 Detail Requirement Description.....	4
2.2.1 Functional Requirements.....	4
2.2.2 Non-functional Requirement.....	5
2.3 Changes of Backlog.....	5
2.4 Iteration of stories.....	5
3. Analysis and Design.....	6
3.1 Identify Entity, Boundary, Control Classes.....	6
3.1.1 Entity Classes.....	6
3.1.2 Control Classes.....	7
3.1.3 Boundary Classes.....	7
3.2 Identify Attributes for each Entity Class.....	8
3.3 Code for Reusable and Available.....	9
3.3.1 Re-usability.....	9
3.3.2 Adaptability.....	9
3.4 Design of the Software.....	10
3.5 Design principles Applied.....	10
4. Implementation and testing.....	11
4.1 Integration build plan.....	11
4.2 Testing.....	12
4.2.1 Plan test.....	12
4.2.2 Design test.....	12
4.3 TDD.....	15
Appendix A: Reference.....	16
Appendix B: Individual Responsibility.....	16
Appendix C: Weekly Reports.....	20
Appendix D: Main screenshots of the system.....	24
Appendix E: Class Diagram.....	33
Appendix F: Junit Test.....	34

1. Project Management

1.1 Introduction

This Software Engineering project is aimed at producing a smart energy management and monitoring system using Java language with Agile methods. This system is used both by energy customers and energy providers. The customers can set budget and get warning if the consumption exceeds the budget, check current price of electricity or gas, check consumption and cost of this month or historical values. The energy providers can check information of customers, set price of electricity or gas, generate and send bills to customers and add or delete customers. This system needs to meet all requirements mentioned in the coursework handout. Our team has 6 members so we use pair programming method. We got together weekly to report progress and discuss problems occurred during this week.

1.2 Agile project management



- At first, we get together and hold the first meeting, everyone wrote down several user stories and we created the product backlog. According to the backlog, our team estimate and iterate plan and workload.
- The sprint period of our team is two weeks
- We decide the sprint backlog on the weekly meeting.
- Sprint Backlog is completed by every member, each member is further assigned by smaller task.
- Cause we can't get together everyday, we set up a WeChat group to assign tasks and communicate every day.
- We held meetings weekly to report progress and discuss problems occurred during this week.
- Every two weeks, we will discuss on the weekly meeting to make sure that the stories in the Sprint Backlog has been fully implemented.
- After making sure that all stories in this Sprint Backlog has fully been achieved, we will start the next iteration.
- Totally, we have 4 iterations in 8 weeks.

1.3 Human resource management

We used pair programming. So, the 6 members are divided into 3 sub-teams with 2 persons in each.

Team number	Member1	Member2	Task
Team 1	Yang Ding	Shiyuan Wang	Analysis, Design, Code and review
Team 2	Wenhao Zhang	Keke Wang	Requirement and Code
Team 3	Jindou Wei	Zhe Wang	Test and Code

1.4 Risk management

1.4.1 Risk Analysis

Risk	Likelihood	Consequence	Solution
The work progress doesn't match the schedule	High	Delay of plans	When making plans, consider possible scenarios that are not in line with reality, and adjust existing plans in time after the occurrence of the situation. The team member should communicate frequently.
Critical errors found in the testing stage.	Medium	Hard to solve	The whole group discussed the errors and spare no effort to test and correct them.
Illness of team member	Medium	The shortage of human resource	Adjust the amount of assignments as soon as possible to balance the burden.
Difficult to pull various parts of project together because of different programming habits	High	The project cannot be Integrated efficiently	Use the same version of IDE, communicate often and integrate the codes often.
Computer crashed during programming	Low	Some important data will lose, and the plan cannot be completed on time	When a part of the task is completed, the existing data and data are backed up and stored in time to prevent data loss due to device breakdown.

1.4.2 Risk planning

- 1) Make the plan flexible so when some members are not available the project still goes on.
- 2) Protect and store the data, backup is necessary.

- 3) Make sure every step in the plan is fully completed and do some analysis before getting into next stage.
- 4) Make back-up plan to avoid emergencies.

1.5 Agile team working

1.5.1 Pair Programming

3 sub-team to do different tasks

1.5.2 Meetings

We have face to face meetings weekly and WeChat meetings daily.

1.6 Schedule and Milestones

Phase	Activity	Star	Duration	Dependencies	Milestone
Prepare	T1. Meeting and sharing	3/19	1		
	T2. Coursework Discussion	3/19	1		
	T3. Fact-finding Techniques	3/18	1	T2	
	T4. Write stories using story cards	3/19-3/20	2	T3	
	T5. Prioritize the stories	3/21	1	T4	
	T6. product backlog	3/21-3/22	2	T5, T4	M1
	T7. Paper prototype user interface	3/22	1		M2
Sprint Cycle, (There were four cycles. The four iterations are the same.)	T8. Deciding on the Sprint Backlog	3/22-3/23	2	T6	
	T9. Estimation and Iteration plan	3/24	1	T8	
	T10. Identify Entity, Boundary and Control classes	3/25	1	T9	
	T11. Identify class relationships	3/35-3/26	2	T10	
	T12. Initial class diagram	3/27-3/29	2	T11	
	T13. Identify attributes for each class	3/29	1	T12	
	T14. Add constraints	3/30	1		
	T15. Develop	3/31-4/1	2	T13, T14	
	T16. Implementation	4/1	1		
	T17. Test and Review	4/1-4/2	2		M3/M4/M5/M6
Final	T18. Beauty of graphical interfaces	5/24-5/25	2	T7	
	T19. Test and Review	5/26	1	T18	M7, Final Software

2. Requirement

2.1 Fact-finding Techniques

2.1.1 Background reading

We carefully read the requirements given by the experiment. From the requirements we know that we are designing a software of a smart energy management and monitoring system. Users of this system can be divided into two types: customers and energy developers. Consumers are referring to families who use this system. They use this system to better control the amount of electricity and gas used at home. Energy suppliers refer to vendors that provide electricity and gas. They need to use this software to better communicate with their customers and manage users.

2.1.2 Observation

In order to understand and design this system better, we observed similar systems around us: Home gas control system, communication operator's user system such as China Unicom. This is an effective way for us to understand the system we need to design. Through observation, we can better understand our needs, understand how users use the system and how the system works.

2.2 Detail Requirement Description

2.2.1 Functional Requirements

- 1) Communication: Smart energy monitoring software can communicate with smart electricity meters and smart gas meters. Energy suppliers can use smart energy management systems to communicate with each home's smart energy monitoring software.
- 2) Real-time information: The system can display the current tariff at any time. Consumers can check their electricity and gas usage at any time, and energy industry can check the amount of consumers. Energy suppliers can modify current price at any time.
- 3) User Management: Energy suppliers can autonomously delete or add users.
- 4) Data records: The system can record the historical consumption of each consumer, and send the total monthly usage to energy suppliers and consumers at the end of each month, and automatically generate bills when sent to consumers.
- 5) User Experience: Customers can set their own monthly budgets, the system will issue an alert after the usage of the month reaches half and the amount of usage in the current month exceeds the budget.

2.2.2 Non-functional Requirement

- 1) Error checking: The software can check the user's wrong operation and issue prompts, such as entering the date format, entering the wrong password, etc.
- 2) Efficiency requirements: The software should be easy to operate and the response time should be as short as possible.
- 3) User-friendly: The user can cancel the operation at any time. Display information to customers in a timely manner during the operation.
- 4) Adaptable: Be able to adapt to changing requirements and can be used in future general markets. For example, connecting to other smart home devices such as smart lights, smart radiators, smart curtains and smart speakers. Software can adapt to future changes. In other words, when developing new software, you should be able to reuse existing components. When adding new features to existing software, you should minimize the impact on existing code.
- 5) Appearance: simple and beautiful interface, keys clear and obvious.

2.3 Changes of Backlog

The backlog that we handed in for the first time was relatively simple, with fewer features listed and ideal for iterative design. The first change was after the first cycle, we just completed the customer code, so we changed the functions of the energy supplier to the second cycle. And after a cycle of work, we have a deeper understanding of this system, so we added some features and stories, such as: the budget can be set as different standards of price or amount; in order to let users can cancel the operation at any time, we added a return button in each interface; after the operation is completed, a prompt box will be displayed to tell the user that the operation is successful, such as logging in or setting budget and tariff. The second change is in the fourth cycle, we had a deeper understanding of the design of the iteration, so after the discussion in the group, we reset each function iteration cycle and target completion date. We put non-functional requirements on the 6th cycle. If the operation is successful, we put the unfinished functional requirements (such as set tariff, bill generation) on the fourth cycle, put the display of user data in the fifth cycle. Also we streamlined the listed functions and stories, we removed the functionality of over design and merged similar stories. We also added notes based on the problem encountered in writing the code.

2.4 Iteration of stories

Iteration	Story	Detail
1	Consumption information	As a customer, I want to know the current electricity and gas consumption that I used.
	Information reception	As an energy provider, I want to receive the consumption of the

		electric or gas every month.
	User information base management	As an energy provider, I want to be able to add or remove customers autonomously.
2	Bills generation	As an energy provider, I want to calculate the bills and send it to customers.
	Cost information check	As a customer, I want to check the current and historical cost of electricity and gas.
	View history	As an energy provider, I want to view the readings and bills history of the consumer.
	Secondary confirmation	As an energy provider, when I want to delete a user's information, I hope the system can pop up a confirmation dialog box.
3	Budget setting and warning	As a customer, I want to set a budget and receive a message if my consumption is above my budget.
	Set tariff	As an energy provider, I want to set and change tariff autonomously.
	Reminder of successful operation	As a user, I hope that if the operation is successful, the system will give me a prompt.

3. Analysis and Design

3.1 Identify Entity, Boundary, Control Classes

There are 3 basic stereotypes in our software system, Entity classes, Boundary classes and Control classes.

3.1.1 Entity Classes

Entity Class are the classes which are used to model information that is long-lived and persistent. Our system has four entity classes. They model the information of User, Provider, System Control, Meter. For example, When customer want to get information of cost and consumption, the system will built an instance of the class.

ControlSys	
removeUser	RemoveUser
sendBill	SendBill
getUserList	GetUserList
changeTariff	ChangeTariff
budget	Budget
addUser	AddUser
billGet	BillGet

Meter	
Meter(String)	
checkCAC()	void
ifMeterExist()	boolean
hi	HistoricalInformation
ID	String
cac	ConsumptionAndCostInfo

Provider	
Provider(String, String)	
getProvider(String, String)	void
genInfo	ArrayList<String>
psw	String
ID	String

User	
User(String, String)	
rfGenInfo()	void
getUser(String, String)	void
genInfo	ArrayList<String>
ID	String

3.1.2 Control Classes

Control Class are used to encapsulate control and coordination of the main actions and control flows. It represents coordination, sequencing, transactions and control of objects.

There are nine control classes in our system. They can control the entity classes. For example, utilizing AddUser.java to add user in the list, and BillGet.java to get bills of customers.

ConsumptionAndCostInfo	
checkCAC(String)	void
tConsumptionAndCostS	String[]
dConsumptionAndCostS	String[]

RemoveUser	
IDExist(String)	boolean
rmID(String)	void
rmIDinList(String)	void

AddUser	
check(String, String)	boolean
addToSys(String, String)	void

HistoricalInformation	
checkData(String, String, String)	int
rDataSList	String[]

ChangeTariff	
readT()	void
tariff	String[]

GetUserList	
userList	String[]

Budget	
getBudget(String)	ArrayList<String>
setBudget(ArrayList<String>, String)	void

SendBill	
ifSend()	boolean
sendNewBill()	boolean

BillGet	
ifNewBill(String)	boolean
billListGet(String)	String[]

3.1.3 Boundary Classes

Boundary Class are used to model the interaction. These often involve receiving (presenting) information and requests from (and to) users and external systems. Normally, Boundary Class represents abstractions of windows, forms, communication interfaces, printer interfaces, sensors, terminals, etc.

HomePage	
HomePage()	
setHomePage()	void
userLogB	JButton
home	JPanel
epLogB	JButton

EPUserInfoPage	
EPUserInfoPage()	
setEPUserInfoPage()	void
hbB	JButton
text	JLabel
cacB	JButton
epUserInfo	JPanel
ID	String
hiB	JButton

LogPage	
LogPage()	
setLogPage()	void
clearB	JButton
login	JPanel
nameText	TextField
pswText	JPasswordField
loginB	JButton

USetBudgetPage		
m	USetBudgetPage()	
m	setUSetBudgetPage()	void
p	setEBDone	JLabel
p	newBText1	TextField
p	newBText2	TextField
p	newBText2inC	TextField
p	uSetBudget	JPanel
p	setBudgB	JButton
p	cBudget	JLabel
p	newBText1inC	TextField
p	setGBDone	JLabel
p	setBudgBinC	JButton

EPSetTariffPage		
m	EPSetTariffPage()	
m	getepSetTariff()	JPanel
m	setEPSetTariffPage()	void
p	newTText1	TextField
p	newTText2	TextField
p	cTariff	JLabel
p	setGTDone	JLabel
p	setETDone	JLabel
p	setTariffB	JButton

UHisInfoPage		
m	UHisInfoPage()	
m	setUHisInfoPage()	void
p	title	String[]
p	uHisInfo	JPanel
p	eDateText	TextField
p	hisInfoSP	ScrollPane
p	sDateText	TextField
p	hisInfoCheckB	JButton
p	hisInfoTable	JTable

UFunctionPage		
m	UFunctionPage()	
m	setUFunctionPage()	void
p	uFunction	JPanel
p	sbB	JButton
p	hbB	JButton
p	ctB	JButton
p	cacB	JButton
p	hiB	JButton

UConsumpAndCostPage		
m	UConsumpAndCostPage()	
m	setUConsumpAndCostPage()	void
p	tEConsumpAndCost	JLabel
p	dEConsumpAndCost	JLabel
p	tGConsumpAndCost	JLabel
p	dGConsumpAndCost	JLabel
p	uConsumpAndCost	JPanel

EPAddUserPage		
m	EPAddUserPage()	
m	setEPAddUserPage()	void
p	epAddUser	JPanel
p	nameText	TextField
p	addUB	JButton
p	pswText	PasswordField
p	pswTextCheck	PasswordField

SEMMS		
m	main(String[])	void
m	startSEMMS()	void
m	setMainPage()	void
m	setBackPage()	void
m	checkLogin(String, String, String)	String
m	addListener()	void
m	addTimer()	void

EPFunctionPage		
m	EPFunctionPage()	
m	setEPFunctionPage()	void
p	ctB	JButton
p	umB	JButton
p	imB	JButton
p	epFunction	JPanel

UHisBillPage		
m	UHisBillPage()	
m	setUHisBillPage()	void
p	uHisBill	JPanel
p	hisBillSP	ScrollPane
p	title	String[]
p	hisBillTable	JTable

EPRemoveUserPage		
m	EPRemoveUserPage()	
m	setEPRemoveUserPage()	void
p	nameText	TextField
p	epRemoveUser	JPanel
p	removeUB	JButton

EPUserManagPage		
m	EPUserManagPage()	
m	setEPUserManagPage()	void
p	ruB	JButton
p	auB	JButton
p	epUserManag	JPanel

EPInfoManagPage		
m	EPInfoManagPage()	
m	setEPInfoManagPage()	void
p	nameText	TextField
p	epInfoManag	JPanel
p	checkUB	JButton

WarningDialog		
m	WarningDialog(JFrame, String, String, String)	
m	WarningDialog(JFrame, String, String)	
p	buttonA	JButton
p	buttonB	JButton
p	text	String
p	warningDia	JDialog

UCheckTariffPage		
m	UCheckTariffPage()	
m	setUCheckTariffPage()	void
p	tariffToStringE	JLabel
p	tariffToStringG	JLabel
p	uCheckTariff	JPanel

3.2 Identify Attributes for each Entity Class

User

Name	Type	Initial value
genInfo	ArrayList<String>	N/A
ID	String	N/A

Provider

Name	Type	Initial value
------	------	---------------

genInfo	ArrayList<String>	N/A
ID	String	N/A
psw	String	N/A

Meter

Name	Type	Initial value
hi	HistoricalInformation	N/A
ID	String	N/A
cac	ConsumptionAndCostInfo	N/A

ControlSys

Name	Type	Initial value
removeUser	RemoveUser	N/A
sendBill	SendBill	N/A
getUserList	GetUserList	N/A
changeTariff	ChangeTariff	N/A
budget	Budget	N/A
addUser	AddUser	N/A
billGet	BillGet	N/A

3.3 Code for Reusable and Available

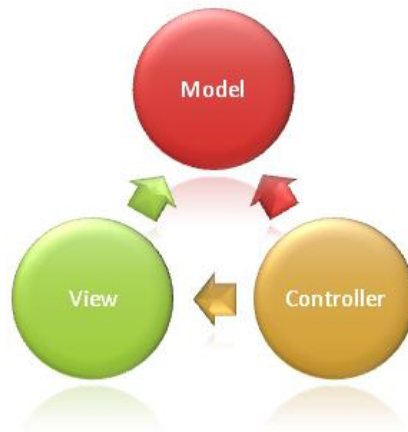
3.3.1 Re-usability

Re-usability provide us a lot of advantages. Instead of writing new repetitive codes, developers can avoid tedious codes and improve the developing efficiency. Our program pay attention on this technique when designing the system. Considering that we use GUI for user convenience and perceptual intuition, there are some warning dialog. When we need to use a warning, we can use a constructor named Warning dialog, so that we don't need to write another method every time.

3.3.2 Adaptability

Our system has good adaptability to different systems. We develop the source code using intelliJ, but it can also be run in cmd successfully.

3.4 Design of the Software



In high-level terms, the MVC pattern means that an MVC application will be split into at least three pieces:

- Models, which contain or represent the data that users work with. These can be simple view models, which just represent data being transferred between views and controllers; or they can be domain models, which contain the data in a business domain as well as the operations, transformations, and rules for manipulating that data.
- Views, which are used to render some part of the model as a UI.
- Controllers, which process incoming requests, perform operations on the model, and select views to render to the user.

As you can see in the Class Diagram, we divided our program into those three parts.

- Models: Provider.java, User.java, Meter.java, ControlSys.java
- Views: HomePage.java, USetBudget.java, UFunction.java, SEMMS.java, EPRemoveUser.java, WarningDialog.java, EPUserInfoPage.java, EPSetTariffPaage.java, UConsumptionAndCostPage.java, EPFunctionPage.java, EPUserMANagPage.java, UCheckTariffPage.java, LogPage.java, UHisInfoPage.java, EPAddUserPage.java, UHisBillPage.java, EPInfoManagPage.java
- Controllers: ConsumptionAndCostInfo.java, HistoricalInformation.java, Budget.java, RemoveUser.java, AddUser.java, ChangeTariff.java, SendBill.java, GetUserList.java, BillGet.java

3.5 Design principles Applied

• Single Responsibility Principle(SRP)

Every object in our smart energy management and monitoring system meet the request to carry on single responsibility. There are only one reason for the class to change. We do not use conditions to judge the carry out in the upper class. Instead our system separates them into different class to meet the SPR.

• Open-Close Principle (OCP)

Software modules should be open for extension and closed for modification. Our codes are friendly to extend. We declared customers into an array list, if there is a new customer, we just need to add him to the list. There may be an infinite extension theoretically.

- **Don't Repeat Yourself Principle (DYP)**

We use generalization methods to reuse our existing code rather than copy them. In our code, we write methods. We use methods when we need same functions so that we don't repeat ourselves.

- **Liskov Substitution Principle (LSP)**

In our system, we use inheritance to simplify the code of our system. In case of the dangerous to change the assumption. We use the abstract method in the superclass to minimize the collision that caused by override. After that we add detail to specific the corresponding method in the subclass.

4. Implementation and testing

4.1 Integration build plan

Version 1

- **Functionality:** The database of the system is formed and the system can read the unit price, the consumption record of day and month(only the last line of database). And then the system can change the data and stores the new data in the files. The system can also look the data in the screen.
- **Use cases:** consumption and cost, Check tariff, Information Checking, User Management
- **Scenarios:** Both the user and the EP can view the database
- **Components:** ConsumptionAndcostInfo.java, Meter.java, Historical information.java, Budget.java, ControlSys.java, User.java

Version 2

- **Functionality:** More function of managing the database, the history data can be analyzed by the system, the system can also set budgets and can inform the user if reaches the budget.
- **Use cases:** History information, Set your tariff
- **Scenarios:** The user can choose a specific date's data and print it in the screen. The user can set the budget and receive message from the system if reaches the budget.
- **Components:** Historical information.java, Meter.java, Budget.java, ControlSys.java, WarningDialog.java

Version 3

- **Functionality:** Improve the capacity of GUI, and the EP can access the users' database and analyze them.
- **Use cases:** All cases that in GUI
- **Scenarios:** Both the user and the EP in the system can go back to the last page or log out to the main page. The EP can view the user's data when EP login the system.
- **Components:** Logpage.java, HomePage.java, EPfunctionPage.java, ConsumptionAndcostInfo.java, Meter.java, EPinfomanagePage.java,

UserInfoPage.java, ControlSys.java, Meter.java, GetUserList.java, UCheckTariffPage.java, UConsumpAndCostPage.java, UFunctionPage.java, UFunctionPage.java, UHisBillPage.java, UHisBillPage.java, UHisInfoPage.java

Version 4

- **Functionality:** The EP can further manage the user's database, it can both add or delete user. The EP can also change the tariff in the database.
- **Use cases:** Changing tariff, User Management
- **Scenarios:** The EP can delete or add users and can change the tariff stored in the file.
- **Components:** ControlSys.java, BillGet.java, changeTrriff.java, EPsettarripage.java, AddUser.java, ControlSys.java, EPUsermanage.java, RemoveUser.java, ControlSys.java, removeUserpage.java, EPUsermanage.java, EPadduserpage.java

4.2 Testing

In the test stage, we use White-box testing to test our system. In each implement step, we use the test to make sure that the function and the purpose can be achieved, after implement stage., the whole system will be test according to the requirements of our plan.

4.2.1 Plan test

We do the test during the implement period, and test some cases after the implementation. We should test the software by component testing and system testing.

- **Component testing:** User Login, Look for the current consumption and cost, Set the budget, History Information, History Bill, Check tariff, Information Checking Energy Provider Login, Add&Remove User, Change Tariff
- **System testing:** update the information and stores in the file.
- **Testing Strategy:** The White-box testing is used in Component testing. And the Black-box testing is used in system testing
- **Success criteria:** 90% of test cases passed. No high priority defects unresolved.

4.2.2 Design test

a. Component Test Cases(using white-box test)

	Test Cases	Case1(correct input)	Case2(incorrect input)	
1	Input	If your username and your password are identical with the data in your	Username and format are wrong(not exist in the database)	The username or password in wrong format

UserList.txt				
2	Result	Login succeed, and the user can select the function	The system will print "Wrong user or password, please try again"	The system will print "Wrong user or password, please try again"
3	Conditions	No conditions exist	No conditions exist	

User login

(just a part of the Component in the real design)

Login	Test description	Test Cases	Samples Pass/Fail	No. Of Bugs	Bug#	Comments
N/A	Input two user login[username-11111, password 11111111], [username-ttt,password rt555]	setup	\	\	\	\
1.1	Test the username 11111 that exists in the database, but the password is empty	1.1	F	0	0	The page not changes. You can't jump to next page
1.2	Test the password 11111111 that is correct for 11111	1.2	p	0	0	Correct behavior
1.3	Test the password 456 that is incorrect for 11111	1.3	F	0	0	The login doesn't success because of the wrong password (print "Wrong user name or password, please try again")
1.4	Test the username and password	1.4	F	0	0	The login doesn't success because of the wrong password (print

1.5	that is not exists in the system Test input empty user name and a password	1.5	F	0	0	"Wrong user name or password, please try again") The page will not change, can not jump to the next page
-----	---	-----	---	---	---	---

b.System Test Cases(using black-box test)

EP requirement: As an energy provider, I want to set and change tariff autonomously, so that I can maintain my profits when costs change.

Method: Scenario-based testing

Test Cases		Case2.1(correct input)	Case2.2(incorrect input)
1	Input	The number of electricity or gas or both of them is imputed in correct format (integer)	Input Enter number in wrong format (not integer)
2	Result	The number of current tariff is changed and the data in database(genInfo.txt) is changed	Result A message "Error: electricity tariff should be a integer larger than 0" is printed. No page switch, and the database remains unchanged.
3	Conditions	No conditions exist	Conditions No conditions exist

Login	Test description	Test Cases	Samples Pass/Fail	No. Of Bugs	Bug#	Comments
N/A	First we can enter the EP's "change Tariff" function and enter the number to set tariff.	setup	\	\	\	\
2.1	Test that the number in genInfo.txt is changed	2.1	p	0	0	The useful message that in the first parameter of each line is changed (correct

behavior)

2.2	The database(genInfo.txt) doesn't change.	2.2	F	0	0	The data remains unchanged (correct behavior)
-----	---	-----	---	---	---	---

4.3 TDD

TDD is a mechanism for software developers to design their functions and to detect whether there is something wrong with the function. A test class is needed.

The screenshot shows an IDE with a file explorer on the left containing files like 11111.txt, 11111D.txt, 11111M.txt, 11111TM.txt, 12345.txt, 12345D.txt, 12345M.txt, and 12345TM.txt. The main editor displays the code for the `UserTest` class:

```
13 class UserTest {
14     @Test
15     public void testUser() {
16         User a = new User( ID: "11111", psw: "11111111");
17
18         assertEquals( expected: "11111", a.getID());
19     }
20 }
```

Below the code editor, the test results are shown:

- Tests passed: 1 of 1 test - 43 ms
- Test Results: 43 ms
- UserTest: 43 ms
- testUser(): 43 ms

The process finished with exit code 0.

For example, we test User, We transmit the parameters to the function to create a new user, the `assertEquals()` will invoke the parameters inside the function and detect whether the answers are identical. From the left of the screen, we can see that the function can match the function. TDD concentrates on unit checking, so that the error can be detected, and not affect the whole system. Junit is a simple unit-testing instrument to support TDD.

Appendix A: Reference

- Chapter 2 – “Software Engineering” lecture by Gokop Goteng in qmplus
- Chapter 3 – “Software Engineering” lecture by Gokop Goteng in qmplus
- <http://www.mountangoatsoftware.com/agile/scrum/scrum-tools/product-backlog/example>
- “Head First OO Analysis & Design” textbook by Brett McLaughlin et al
- https://en.wikipedia.org/wiki/Software_design
- Chapter 4 – “Software Engineering” lecture by Gokop Goteng in qmplus
- Chapter 5 – “Software Engineering” lecture by Gokop Goteng in qmplus
- “Head First Software Development” textbook by Dan Pilone, Russ Miles
- Agile Java™: Crafting Code with Test-Driven Development, Jeff Langr
- Abran, Alain; Moore, James W.; Bourque, Pierre; Dupuis, Robert; Tripp, Leonard L. (2004). Guide to the Software Engineering Body of Knowledge. IEEE. ISBN 0-7695-2330-7.
- Sommerville, Ian (2008). [Software Engineering](#) (7 ed.). Pearson Education. ISBN 978-81-7758-530-8. Retrieved 10 January 2013.
- https://en.wikipedia.org/wiki/Software_testing
- <https://www.ox.ac.uk/admissions/graduate/courses/msc-software-engineering?wssl=1>

Appendix B: Individual Responsibility

Summary of Responsibilities and Achievements

Group Members

Name	QMID	BUPTID	CLASS
Wenhao Zhang	151010671	2015213067	2015215108
Yang Ding	151010888	2015213088	2015215109
Keke Wang	151011069	2015213106	2015215109
Jindou Wei	151011416	2015213140	2015215110
Zhe Wang	151011391	2015213138	2015215110
Shiyuan Wang	151011324	2015213132	2015215110

Group leader: Wenhao Zhang

Individual Members Contribution and Self-Appraisal of Work done

Individual member: Wenhao Zhang

(1) Individual member contribution

Zhang Wenhao is the Group Leader in the group and the Scrum Master in the

Project. He is responsible for the coordination of the whole team. He also motivates the whole group, arranges daily WeChat communication, holds meetings weekly. Every time before a meeting, he informs the time and location through WeChat. He also wrote many codes to achieve functions of the system. He takes up new ideas actively and shared them with group members. He is responsible for 7 parts of backlog. The story ID is c3, c4, c5, c6, c8, c9, c10. The user manual and project management part of report is also created by him.

(2) The outcome appraisal of the work

As the group leader, Wenhao Zhang is good at management and he tries his best to help the group members, motivate the whole group and arrange tasks for the members. He is warm-hearted and he helps to solve problems during programming. He also played a role in analysis and design. He contributes a lot to write the codes. He achieves many functions of the system, such as the function of setting budget, checking consumption and cost of this month, deleting user account, etc. Wenhao Zhang is a good team leader who arranges the group well and meanwhile completing his tasks well.

(3) Self-appraisal of work done

During the project, I tried my best to motivate the whole group, arrange meetings and tasks and help the group members to make sure that they can complete their tasks and can work together to achieve the same goal. I think I learned a lot through this project. For example, I learned how to work with others to achieve the same goal. I learned how to make arrangement for a team to make the team run correctly. I also learned how to write better codes to make the whole system to be more stable. I have a deeper understanding of what we learned from our teacher, such as agile project management, human resource management and risk management.

Individual member: Yang Ding

(1) Individual member contribution

Be in the discussion of improving the backlog in detail of every part of it. Working on all part of program. Design the structure of program, combine the control classes from other member and modified it to fit the need of objective oriented programming and improve it in further development of program. Finish 90 percent of Graphic User Interface on his own. Working on the java doc after all part of program is done.

(2) The outcome appraisal of the work

As the most skillful programmer in the group, Yang Ding put most of his effort in to working on the program. With he is doing more programming than others, other members can focus more on the remaining part of the coursework, during the meeting, he will share his thought when coming to the part that's related to program function and overall design, we are glad to work with him as a team.

(3) Self-appraisal of work done

During the coursework, as personally I'm not good at backlog and all other parts

except programming than my team member, I contribute a lot in programming and didn't do much in other part. When design the structure and discuss the design with other members, I got deeper understanding in objective oriented programming and the relationship between programmer and other roles in a group. When programming the code, I find out some new useful class in API and got new way of thinking in design a program. During the whole coursework, I do want I good at and try to do it great, but I think there is still more to learn in programming techniques, and I should also try to do more in other part of group work and train my other skills as well next time.

Individual member: Keke Wang

(1) Individual member contribution

Be responsible for 5 parts of backlog, the story ID is e1, e3, e6, e7 and e8.

Draw the prototype in the first checking time, talking with program designer to modify it.

Draw the class diagram in the second time. Comb the relationship among entity class, control class and boundary class.

Working on the function of changing tariff and GUI of checking tariff.

Write this report in Appendix and reference parts.

(2) The outcome appraisal of the work

As the most diligent group member, Keke Wang always devote herself to the work which group leader arranges. She is never late when we have meeting, she gives us idea actively, and do more other things (like prototype and class diagram) than others, with her help, we have less problems.

(3) Self-appraisal of work done

During coursework, I'm good at drawing, so I took the initiative to take the responsibility of prototype and class diagram. When design the structure of class diagram and discuss the design with other members, I got deeper understanding in objective oriented programming and the relationship between programmer and other roles in a group which help me drawing, then I want to try some codes in program, due to the help of our group members I finished it on time. I love my group, and thanks them a lot.

Individual member: Jindou Wei

1) Individual member contribution:

Jindou Wei is the team102's member. She Worked with other group members to writing the backlog and Paper Prototype. She response for the Analysis and design part and wrote that part in the final report. She also did the billget function and warning dialog page in the code.

2) The outcome appraisal of the coursework

Jindou Wei is a very kind person and always positive in group discussion. In every group meeting, she can express her view actively, and finish the task as

requested on her own. She is definitely a good team member to work with.

3) Self-appraisal of work done

During the whole period of software engineering coursework, I learned a lot about how to engineer a project from zero to all. Although there are some challenge for me, I never give up and try my best to figure out the questions. To finish my task efficiently, I reviewed the lectures many times and get a better understanding on Software Engineering. Also, I learned a lot about how to work in a team. I really appreciate my group member. They not only help me with the class knowledge and the coding but also teach me how to share workloads and new idea with my partner.

Individual member: Zhe Wang

(1) Individual member contribution

Be responsible for 6 parts of backlog, the story ID is e2, e4, e9 e8, c9 and c10

Draw the prototype in the first checking time.

class, control class and boundary class.

Working on the function of get user list and add user.

Write the requirement part of report.

(2) The outcome appraisal of the work

Zhe Wang is not good at writing code, so she is only responsible for writing some of the code outside the GUI. She is actively looking for information for team members and carefully completing the work she can do, such as drawing and writing reports.

(3) Self-appraisal of work done

As a member of the agile team for this software development, I seriously completed my work and learned a lot about software development during this process. During each phase, I applied this knowledge to practice and collaborated with my team members to benefit from it. During the weekly meetings, we can exchange our ideas and listen carefully to others. Thanks to my team members for helping me and contributing to this course.

Individual member: Shiyuan Wang

(1). Individual member contribution:

Shiyuan Wang contributes some good ideas to construct the the system. He developed some functions to control the data in the file, Which is for generating the data for GUI to view. some of the functions that achieved by him has been added in the system(SEMMS.java, SendBill.java). After finishing each iteration, he tested the component to make sure that their is no error and the function can achieve the desired function, When the system finishes, he did tests to make sure that the system's capacity can meet the backlog.

(2). The outcome appraisal of the coursework

As a student in our team, Shiyuan Wang is a person that is very careful and can

always find the problem, he solved many problems in the programming and testing procedure. He is willing to learn new knowledge and is responsible for the tasks he did. His patience helped us to overcome many bugs and avoided many structural weaknesses.

(3). Self-appraisal of work done

At the beginning, I had many problems and had no macro planning about the program, after the discussions in our group, I began to have ideas and started programming. My teammates taught me to construct a general structure of a relatively bigger program, and I learned to have bigger ideas and broke them into small parts, solved small problems and cares about the relationship and interface to others' program.

Appendix C: Weekly Reports

Week1

- **Time:** 2018.3.21 Wednesday 19:00

- **Location:** Man coffee in school

- **Conference content:**

Everyone is familiar with each other

Understand the advantages and disadvantages

Select leader

Arrange the coursework of everyone

Take the group photo

- **Task:**

Give main idea of program code

Have an arrangement of coursework

Week2

- **Time:** 2018.3.30 Friday 18:30

- **Location:** Man coffee in school

- **Attendance content:**

Arrange the code

Check the completion

Talk about some important problem

Take the group photo

- **Task Completion:**

Half of backlog

The code of cost information and consumption information about customer.

Half of prototype

- **Task:**

Complete backlog in detail

Continue drawing prototype

Week3

- **Time:** 2018.4.8 Thursday 18:30
- **Location:** Man coffee in school
- **Attendance content:**
Arrange the left code
Check the completion
Talk about some important problem
Take the group photo
- **Task Completion:**
Half of backlog
The code of price information about customer.
Prototype
- **Task:**
Complete backlog in detail
Complete prototype in detail
Complete the code about customer

Week4

- **Time:** 2018.4.12 Monday 13:30
- **Location:** The third floor of school building
- **Attendance content:**
Arrange the code
Check the completion
Talk about some important problem
Take the group photo
- **Task Completion:**
Backlog
The code of interface about customer.
Half of prototype
- **Task:**
Complete backlog in detail
Continue drawing prototype
Complete the code about customer

Week5

- **Time:** 2018.4.22 Sunday 10:00
- **Location:** Man coffee in school
- **Attendance content:**
Arrange the code
Check the completion
Talk about some important problem

Take the group photo

- **Task Completion:**

The code of historical information and budget setting about customer.

The code of warning in interface

Prototype

- **Task:**

Complete the code about customer

Week6

- **Time:** 2018.4.30 Monday 19:00

- **Location:** The third floor of school building

- **Attendance content:**

Arrange the code

Check the completion

Talk about some important problem

Take the group photo

- **Task Completion:**

The code of information checking, bill generation and tariff setting about provider.

The code of the button action listener about GUI

Half of diagram

- **Task:**

Complete the code about provider

Complete the class diagram in detail

Complete the report of program

Week7

- **Time:** 2018.5.8 Tuesday 19:00

- **Location:** Man coffee in school

- **Attendance content:**

Arrange the code

Check the completion

Talk about some important problem

Take the group photo

- **Task Completion:**

The code of add user about provider.

The code of the secondary confirmation about GUI

Diagram

- **Task:**

Complete the code about provider

Complete the report of program

Week8

- **Time:** 2018.5.18 Tuesday 19:00
- **Location:** The forth floor of school building
- **Attendance content:**
Arrange the code
Check the completion
Talk about some important problem
Take the group photo
- **Task Completion:**
The code of add user about provider.
The code of the secondary confirmation about GUI
Half of report
- **Task:**
Complete the code about provider in detail
Write the report of program in detail

Week9

- **Time:** 2018.5.25 Friday 19:00
- **Location:** Man coffee in school
- **Attendance content:**
Arrange the code
Check the completion
Talk about some important problem
Take the group photo
- **Task Completion:**
The code of remove user about provider.
The code of GUI for provider
Report
- **Task:**
Complete the code about provider in detail
Complete the report in detail

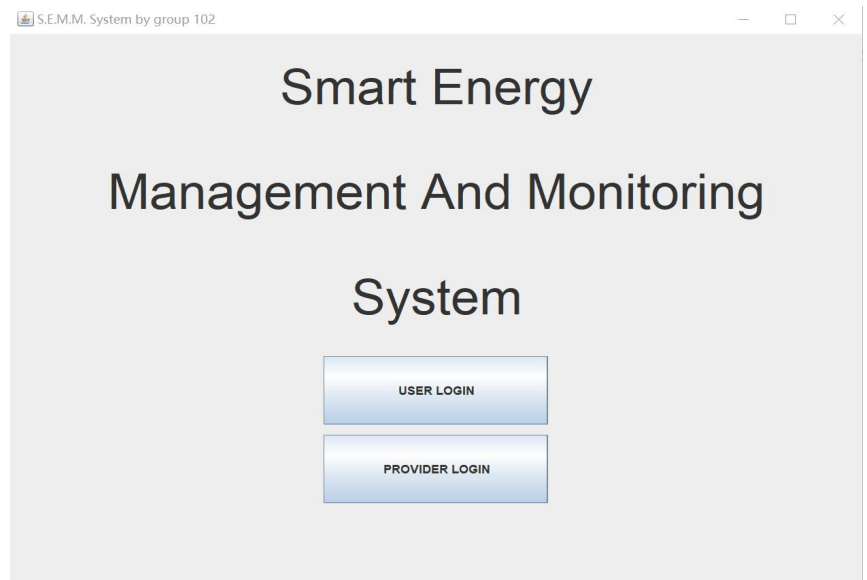
Week10

- **Time:** 2018.5.28 Sunday 14:00
- **Location:** Man coffee in school
- **Attendance content:**
Talk about some important problem
Take the group photo
- **Task Completion:**
All completed
- **Task:**

Complete the code about provider in detail
Complete the report in detail

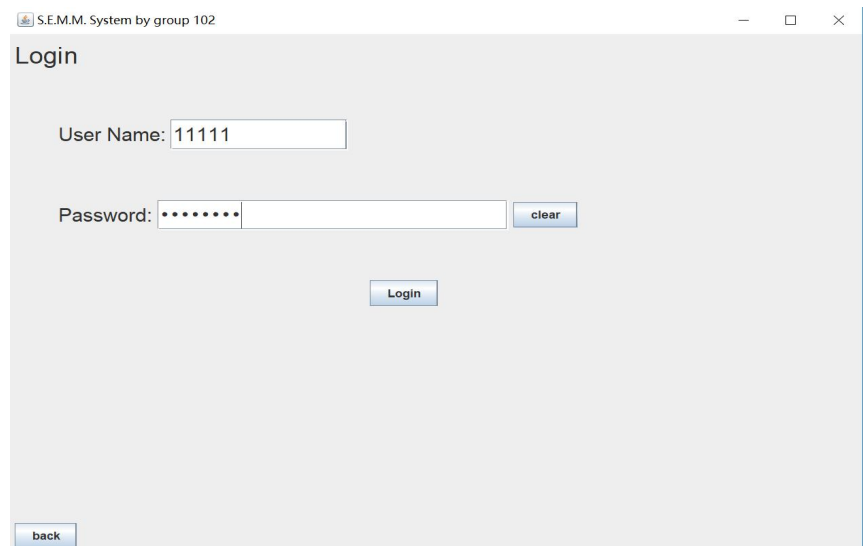
Appendix D: Main screenshots of the system

Main page of the system.



User_login

Enter the right name and password, otherwise it will remind you "input invalid".



User
Functional button
Click one to see more
details.

S.E.M.M. System by group 102

Please choose the function you want:

- Consumption and Cost
- Set Your Budget
- History Information
- History Bill
- Check Tariff

logout

Consumption&Cost
Show the consumption
and cost of today and
since last change.
Update information
every 30s.

S.E.M.M. System by group 102

Consumption and cost

Today	consumption	cost
Electricity:	2.12kwh	£ 11.63
Gas:	1.72m^3	£ 6.24

Since last charge	consumption	cost
Electricity:	10.62kwh	£ 181.63
Gas:	3.42m^3	£ 57.24

back

Set Budget
The user can set new
budget in cost or in
usage by entering a
number in the field.

S.E.M.M. System by group 102

Set budget

	electricity	gas
Current budget:	£ 300(300.00kwh)	£ 200(100.00m^3)
New budget(in cost):	£ <input type="text"/>	£ <input type="text"/> <input type="button" value="OK"/>
New budget(in usage):	<input type="text"/> kwh	<input type="text"/> m^3 <input type="button" value="OK"/>

Enter new budget in right line (pounds or kwh/m^3) if you want to change any of your current budget, and then click 'OK'

back

History information

Click check button, user can view history information in excel.

S.E.M.M. System by group 102

History information

Start date:

Expiratoin date:

Date	Electricity Consumption(kwh)	Electricity Cost(£)	Gas Consumption(m³3)	Gas Cost(£)
------	------------------------------	-----------------------	----------------------	---------------

S.E.M.M. System by group 102

History information

Start date:

Expiratoin date:

Date	Electricity Consumption(kwh)	Electricity Cost(£)	Gas Consumption(m³3)	Gas Cost(£)
18.2.20	0.5	10.00	0.1	3.00
18.2.21	0.5	10.00	0.1	3.00
18.2.22	0.5	10.00	0.1	3.00
18.2.23	0.5	10.00	0.1	3.00
18.2.24	0.5	10.00	0.1	3.00
18.2.25	0.5	10.00	0.1	3.00
18.2.26	0.5	10.00	0.1	3.00
18.2.27	0.5	10.00	0.1	3.00
18.2.28	0.5	10.00	0.1	3.00
18.3.1	0.5	10.00	0.1	3.00
18.3.2	0.5	10.00	0.1	3.00
18.3.3	0.5	10.00	0.1	3.00
18.3.4	0.5	10.00	0.1	3.00
18.3.5	0.5	10.00	0.1	3.00
18.3.6	0.5	10.00	0.1	3.00
18.3.7	0.5	10.00	0.1	3.00
18.3.8	0.5	10.00	0.1	3.00
18.3.9	0.5	10.00	0.1	3.00
18.3.10	0.5	10.00	0.1	3.00
18.3.11	0.5	10.00	0.1	3.00
18.3.12	0.5	10.00	0.1	3.00
18.3.13	0.5	10.00	0.1	3.00

History Bill

History bill will show when click the Functional button: history bill.

S.E.M.M. System by group 102

History Bill

Charge Date	Electricity Consumption(kwh)	Electricity Cost(£)	Gas Consumption(m³3)	Gas Cost(£)
18.2.28	14.0	280.00	2.8	54.00
18.3.31	15.5	310.00	3.1	93.00
18.4.30	15.0	300.00	3.0	90.00

Check Tariff

Tariff will show when click the Functional button: Check tariff.

S.E.M.M. System by group 102

Check tariff

Current tariff: electricity: £ 1.0/kwh

gas: £ 2.0/m^3

back

Click logout button that the user will be back to the main interface and then exit.

S.E.M.M. System by group 102

Please choose the function you want:

Consumption and Cost

Set Your Budget

Are you sure you want to logout?

YES NO

History Bill

Check Tariff

logout

Provider_login

The provider must input right name and password, otherwise it will remind you “input invalid” .

S.E.M.M. System by group 102

Login

User Name:

Password: clear

Login

back

Provider
Functional button
Click one to see more
details.

S.E.M.M. System by group 102

Please choose the function you want:

User Managent

Information Checking

Change Tariff

logout

This screenshot shows the main menu of the S.E.M.M. System. It features a title bar with the text 'S.E.M.M. System by group 102' and standard window controls. The main content area displays the instruction 'Please choose the function you want:' followed by three blue buttons: 'User Managent', 'Information Checking', and 'Change Tariff'. A 'logout' button is located at the bottom left of the window.

Management
The provider can add or
remove users when click
the button.

S.E.M.M. System by group 102

User Management

Add User

Remove User

back

This screenshot shows the 'User Management' screen. It has a title bar with 'S.E.M.M. System by group 102'. The main content area is titled 'User Management' and contains two blue buttons: 'Add User' and 'Remove User'. A 'back' button is positioned at the bottom left.

Add User
The provider must enter
the name whose do not
exist and set a password
for user.

S.E.M.M. System by group 102

Add User

User Name:

Password:

Password check:

OK

back

This screenshot shows the 'Add User' screen. It features a title bar with 'S.E.M.M. System by group 102'. The main content area is titled 'Add User' and contains three text input fields: 'User Name:', 'Password:', and 'Password check:'. Below the fields is an 'OK' button. A 'back' button is located at the bottom left.

Add user
Add successfully.

S.E.M.M. System by group 102

Add User

User Name:

Password:

Password check:

User has been added

OK

OK

back

Remove User
The provider must enter
the name whose that
exists.

S.E.M.M. System by group 102

Remove User

User Name:

remove

back

Remove User
Remove successfully.

S.E.M.M. System by group 102

Remove User

User Name:

remove

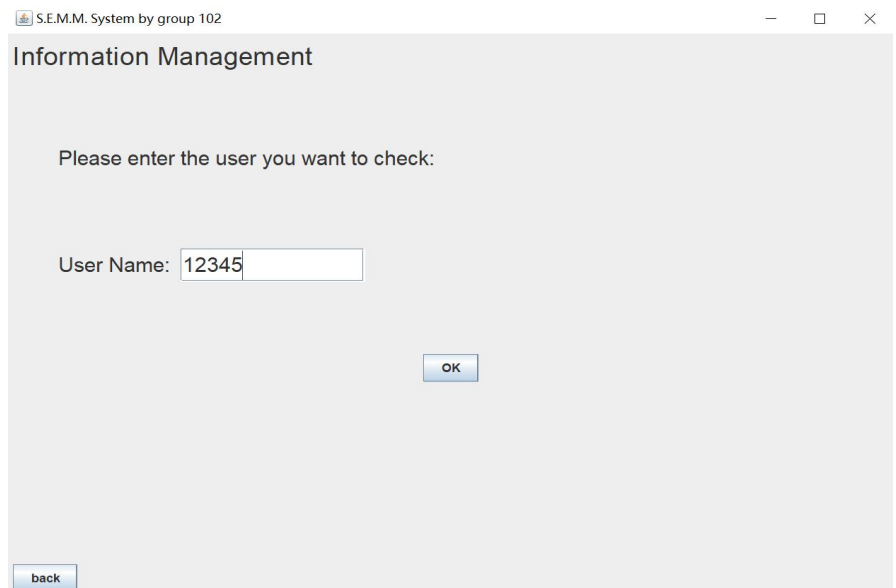
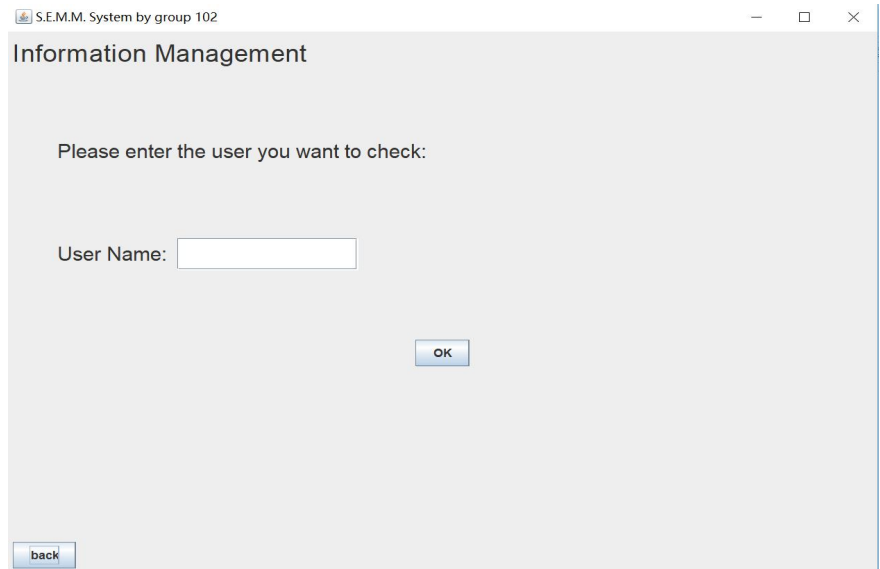
User removed

OK

back

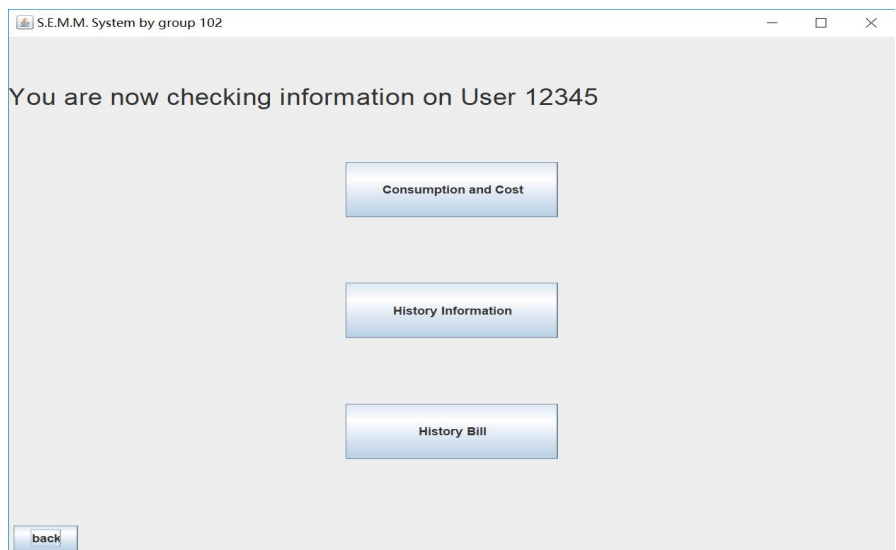
Information_check

Enter right name of user and click “ OK ” , the provider can see more information of user in details.



Information_check

Choose one functional button.



Consumption&Cost

Show the consumption and cost of today and since last change.

Update information every 30s.

History information

Click check button, user can view history information in excel.

S.E.M.M. System by group 102

Consumption and cost

Today	consumption	cost
Electricity:	2.71kwh	£ 12.22
Gas:	2.31m^3	£ 7.42

Since last charge	consumption	cost
Electricity:	11.21kwh	£ 182.22
Gas:	4.01m^3	£ 58.42

[back](#)

S.E.M.M. System by group 102

History information

Start date:

Expiratoin date:

[Check](#)

Date	Electricity Consumption(kwh)	Electricity Cost(£)	Gas Consumption(m^3)	Gas Cost(£)
18.2.24	0.5	10.00	0.1	3.00
18.2.25	0.5	10.00	0.1	3.00
18.2.26	0.5	10.00	0.1	3.00
18.2.27	0.5	10.00	0.1	3.00
18.2.28	0.5	10.00	0.1	3.00
18.3.1	0.5	10.00	0.1	3.00
18.3.2	0.5	10.00	0.1	3.00
18.3.3	0.5	10.00	0.1	3.00
18.3.4	0.5	10.00	0.1	3.00
18.3.5	0.5	10.00	0.1	3.00
18.3.6	0.5	10.00	0.1	3.00
18.3.7	0.5	10.00	0.1	3.00
18.3.8	0.5	10.00	0.1	3.00
18.3.9	0.5	10.00	0.1	3.00
18.3.10	0.5	10.00	0.1	3.00
18.3.11	0.5	10.00	0.1	3.00
18.3.12	0.5	10.00	0.1	3.00
18.3.13	0.5	10.00	0.1	3.00
18.3.14	0.5	10.00	0.1	3.00
18.3.15	0.5	10.00	0.1	3.00
18.3.16	0.5	10.00	0.1	3.00
18.3.17	0.5	10.00	0.1	3.00

[back](#)

S.E.M.M. System by group 102

History Bill

Charge Date	Electricity Consumption(kwh)	Electricity Cost(£)	Gas Consumption(m^3)	Gas Cost(£)
18.2.28	14.0	280.00	2.8	54.00
18.3.31	15.5	310.00	3.1	93.00
18.4.30	15.0	300.00	3.0	90.00

[back](#)

Set Tariff

The provider can set new tariff for electricity and gas by entering the number in the field.

S.E.M.M. System by group 102

Set Tariff

	electricity	gas
Current tariff:	£ 1.0	£ 2.0
New tariff:	£ <input type="text"/>	£ <input type="text"/> <input type="button" value="OK"/>

Enter new tariff if you want to change any of current tariff, and then click 'OK'

Set Tariff

Set successfully.

S.E.M.M. System by group 102

Set Tariff

	electricity	gas
Current tariff:	£ 2.0	£ 4.0
New tariff:	£ <input type="text"/>	£ <input type="text"/> <input type="button" value="OK"/>

Enter new tariff if you want to change any of current tariff, and then click 'OK'

Electricity tariff changed!

Gas tariff changed!

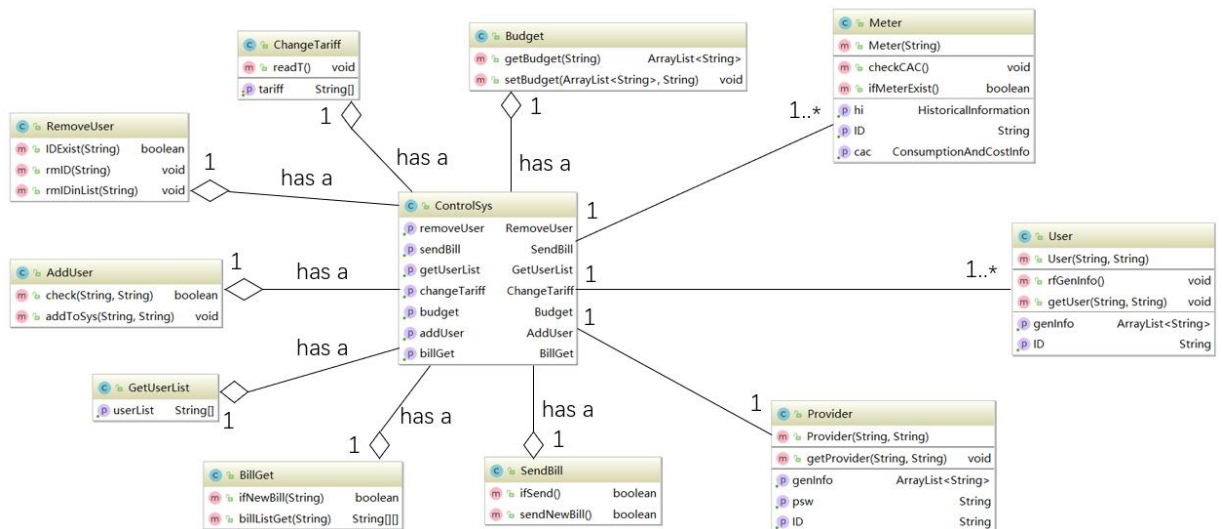
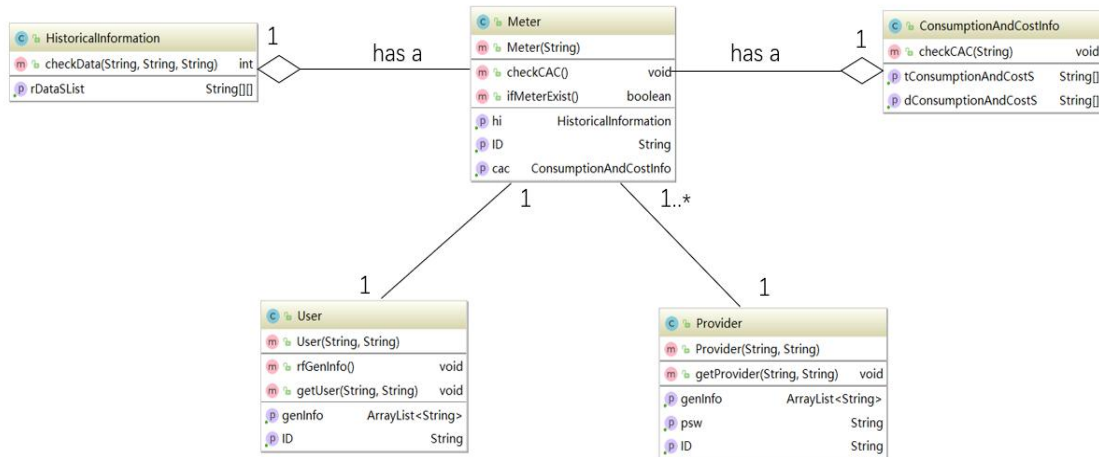
Click logout button that the user will be back to the main interface and then exit.

S.E.M.M. System by group 102

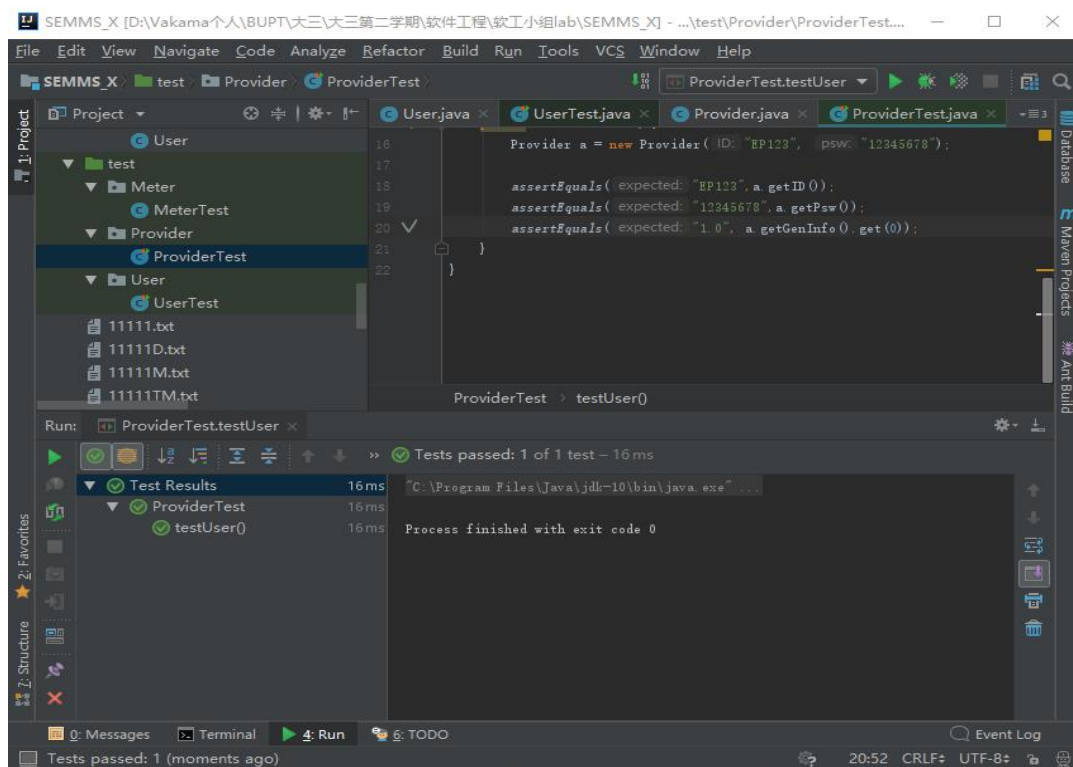
Please choose the function you want:

Are you sure you want to logout?

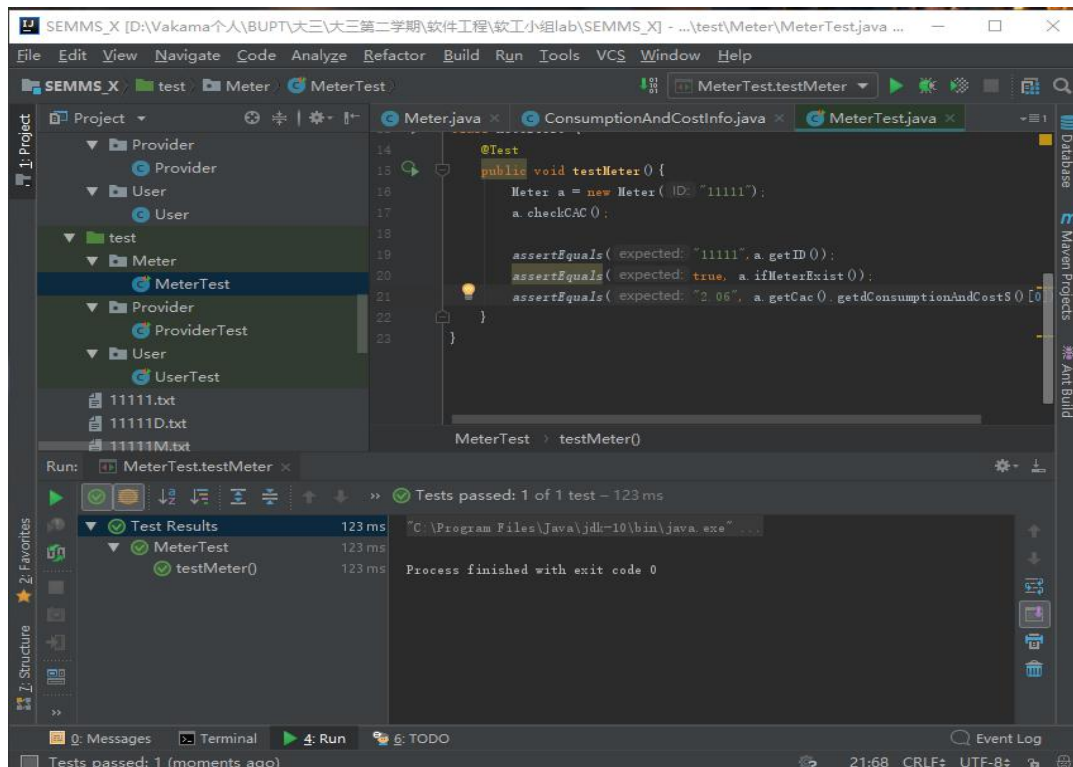
Appendix E: Class Diagram



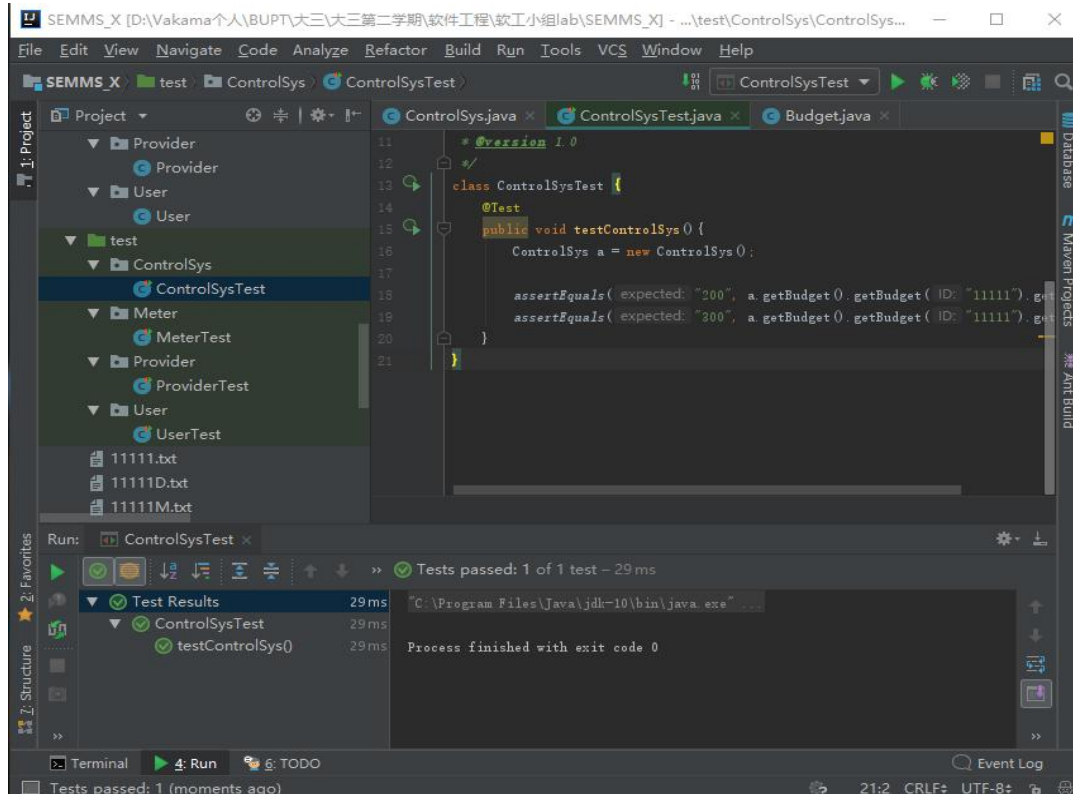
✓ Provider.java



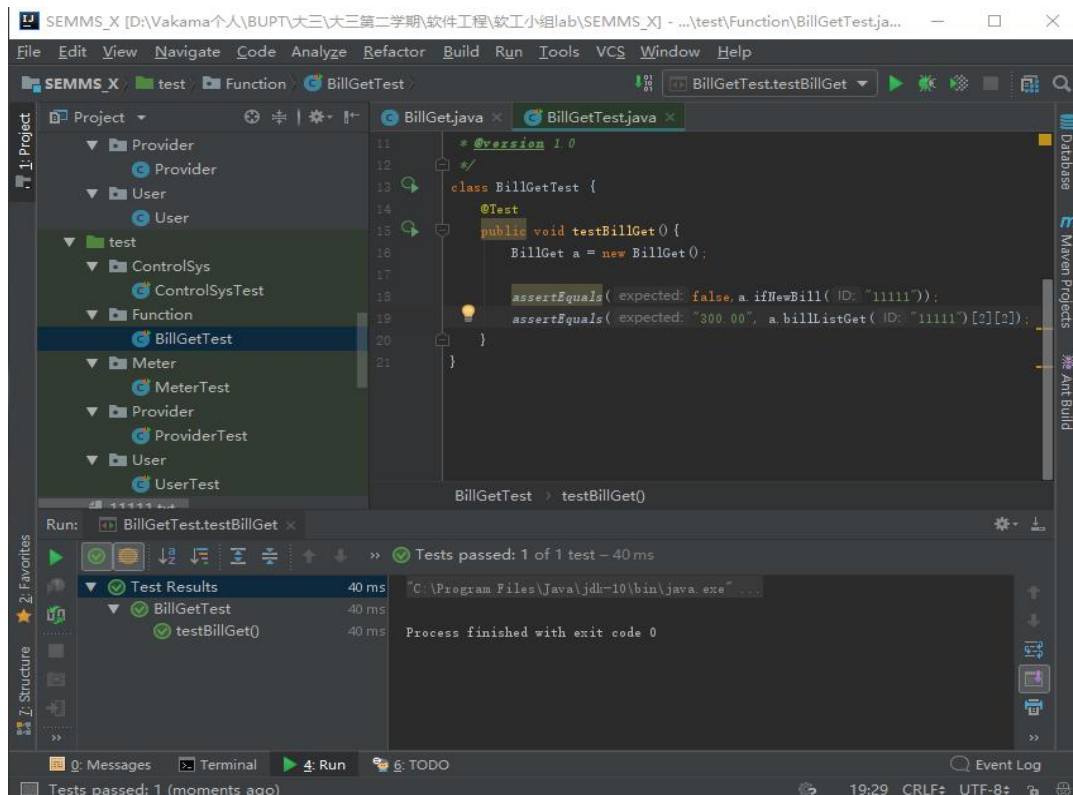
✓ Meter.java



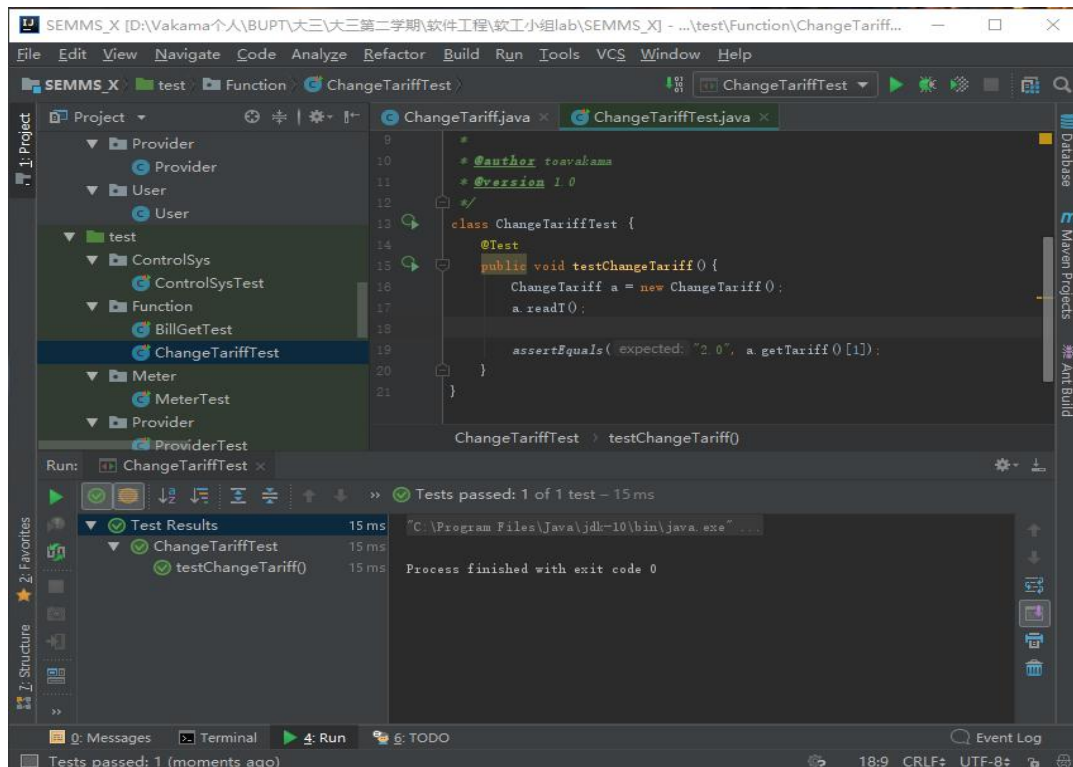
✓ ControlSys.java



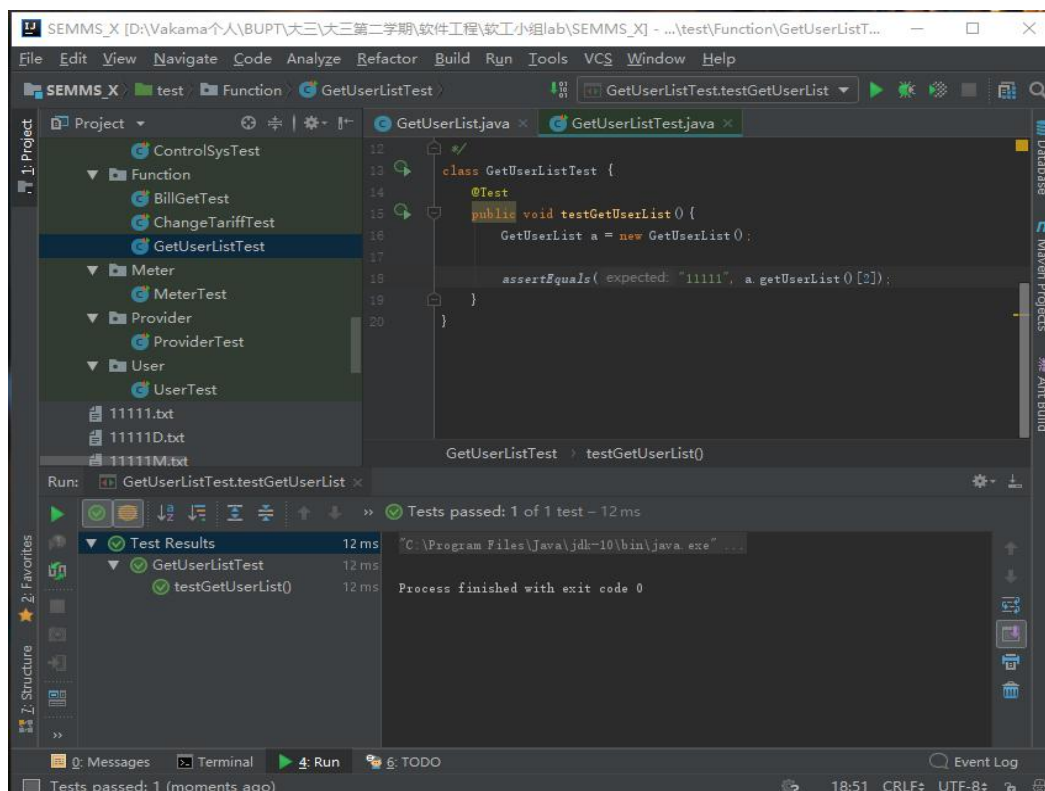
✓ BillGet.java



✓ ChangeTariff.java



✓ GetUserList.java



✓ SendBillTest.java

