



AMBERSTON

HIGH SCHOOL

Project Report

ON

Banking System

Submitted to

AMBERSTON HIGH SCHOOL

in partial fulfilment of the requirements for the final project of

ICS3U

Submitted By

Yiduo Cai	600-609-673
Shuhan Yu	160-791-125
Ruolin Jiang	158-400-135
Mingxi Luo	

Supervised By

Mrs. Namarta Vij

COMPUTER SCIENCE

AMBERSTON HIGH SCHOOL

MARKHAM

[MARCH, 4th]

To whomsoever it may concern

This is to certify that the project report entitled “Traffic Light Management System” was conducted by Shuhan Yu, Yiduo Cai and Ruolin Jiang , both of whom are officially registered students of Amberson High School, completed under my guidance and supervision. They submitted this project in partial fulfillment of the requirements for the degree of Amberson High School.

Their dissertation represents the original work and is worthy of consideration for the award of the credit of Computer Science.

Declaration

I, “Yiduo Cai, Shuhan Yu, Ruolin Jiang ”, hereby declare that the work presented herein is genuine work done originally by me and has not been published or submitted elsewhere for the requirement of a degree program. Any literature, data or works done by others and cited within this dissertation has been given due acknowledgement listed in the reference section.

Yiduo Cai

Shuhan Yu

Ruolin Jiang

Mingxi Luo

Reg No. 600-609-673. 160-791-125. 158-400-135

Date: 2025.2.28

Abstract

This project creates a traffic light management system that will simulate the functionality of a traffic light at an intersection. The system will allow simulation of traffic light cycles, traffic flow, and monitoring of traffic at various intersections. This project will be implemented in Java and will cover concepts such as arrays, ArrayList, classes, inheritance, and encapsulation.

KEY WORDS

Java, Html, Arrays, ArrayList, Classes, Inheritance, and Encapsulation

Table of Content

Chapter-1	6
Introduction about java	6
Introduction to Eclipse	7
Chapter-2	8
Configuration used in project	8
Chapter-3	10
Explanation to project	10
Chapter-4	13
Features and Functionality.....	13
Challenge and Solutions	15
Chapter-5	16
HTML Implementation and Screenshots.....	16
Chapter-6	23
Conclusion	23
Chapter-7	24
Project Allocation	24
Reference.....	25

Chapter-1

Introduction about java

In our project, the main programming language used is Java. Java is a "write once, run anywhere" programming language. It is a class-based object-oriented programming language designed to minimize implementation dependencies. Java was developed by James Gosling in May 1995 and later acquired by Oracle Corporation. It is like a "master key" that can easily adapt to different devices such as computers, mobile phones, servers and even smart home appliances, thanks to its core mechanism - the Java Virtual Machine (JVM), which can translate code into instructions that can be understood by various devices.

Java is known for its simplicity, security and stability. It supports modular development, making code as easy to reuse and maintain as building blocks; it also comes with an "automatic garbage collection" function to avoid memory leaks. Whether it is an App on a mobile phone, a bank's backend system, or an enterprise-level website and big data processing, Java is the main development language. Millions of developers around the world choose it not only because of its powerful functions, but also because of its rich free tools and active community support.

Chapter-1

Introduction to Eclipse

The software we use in the process of writing code is called Eclipse. Eclipse is often used as a teaching tool in computer courses in colleges and universities. Its clear engineering structure and debugging functions are conducive to algorithm practice. Financial institutions and telecom operators also often use Eclipse because it has a highly constructive and reliable system. Eclipse is an open source, cross-platform integrated development environment (IDE). It has become a multi-language programming tool widely used by developers around the world. It is suitable for multiple scenarios such as academic research, commercial applications and open source projects.

The core design of Eclipse is to simplify complexity. When you enter code, it is like a vigilant assistant, which can automatically complete a part of repeated code, check real-time errors, and immediately underline the errors without having to search hundreds of lines to modify missing brackets. It is also a good debugging tool that allows you to pause code execution, check variables, and gradually fix errors.

Secondly, what really stands out about Eclipse is its flexibility. Eclipse Marketplace offers thousands of plugins that turn the IDE into a cockpit tailored to your project needs. Popular add-ons turn complex tasks into simple clicks. Including the ability to work with Python and HTML.

Finally, Eclipse has strong cross-platform and collaborative capabilities. It can run on Windows, Linux, and macOS systems while maintaining functional consistency and supporting team collaborative development mode.

Chapter-2

Configuration used in project

Debbie Cai

Hardware Configuration:

Processor: .1GHz dual-core Intel Core i3, Turbo Boost up to 3.2GHz, with 4MB L3 cache

RAM: 16.0GB (15.5GM usable)

Hardware UUID: F3EE01D1-0585-593D-A460-BF22E8D66677

Provisioning UDID: 00008122-001479180A80001C

Types of Peripheral Devices:

A peripheral device is an internal or external hardware component that connects to a computer to extend its capabilities. Peripheral devices can be purchased after manufacturing to provide input/output (I/O) functions or improve the way the computer carries out specific processing or storage tasks. The types of peripheral: Input peripherals, Output peripherals, Storage peripherals, Net-working peripherals The peripheral I used most: keyboard and monitor.

Shuhan Yu

Hardware Configuration:

Processor: 1GHz dual-core Intel Core i3, Turbo Boost up to 3.2GHz, with 4MB L3 cache

RAM: 16.0GB (15.5GM usable)

Hardware UUID: F2B3D668-75A5-5540-B837-FB1C98DEBFC7

Provisioning UDID: 00008112-000A28D0343BC01E

Types of Peripheral Devices:

A peripheral device is an internal or external hardware component that connects to a computer to extend its capabilities. Peripheral devices can be purchased after manufacturing to provide input/output (I/O) functions or improve the way the computer carries out specific processing or storage tasks. The types of peripheral: Input peripherals, Output peripherals, Storage peripherals, Net-working peripherals The peripheral I used most: keyboard and monitor.

Mingxi Luo

Hardware Configuration:

Processor: 1GHz dual-core Intel Core i3, Turbo Boost up to 3.2GHz, with 4MB L3 cache

RAM: 16.0GB (15.5GM usable)

Hardware UUID: 6C7A02B0-B5BC-5A6C-86C6-A0B3596399A0

Provisioning UDID: 00008112-000E64163621401E

Types of Peripheral Devices:

A peripheral device is an internal or external hardware component that connects to a computer to extend its capabilities. Peripheral devices can be purchased after manufacturing to provide input/output (I/O) functions or improve the way the computer carries out specific processing or storage tasks. The types of peripheral: Input peripherals, Output peripherals, Storage peripherals, Net-working peripherals The peripheral I used most: keyboard and monitor.

Ruolin Jiang

Hardware Configuration:

Processor: 1GHz dual-core Intel Core i3, Turbo Boost up to 3.2GHz, with 4MB L3 cache

RAM: 16.0GB (15.5GM usable)

Hardware UUID: 0080E53A-532A-52E5-9D53-3190BA16C0A0

Provisioning UDID: 00008122-0008715401F1001C

Types of Peripheral Devices:

A peripheral device is an internal or external hardware component that connects to a computer to extend its capabilities. Peripheral devices can be purchased after manufacturing to provide input/output (I/O) functions or improve the way the computer carries out specific processing or storage tasks. The types of peripheral: Input peripherals, Output peripherals, Storage peripherals, Net-working peripherals The peripheral I used most: keyboard and monitor.

Chapter-3

Project Overview

The Banking System project is designed to simulate a basic banking application where users can create accounts, manage funds, and view transaction history. The system is implemented in Java using Object-Oriented Programming (OOP) principles and various data structures (Arrays, ArrayLists, etc.). Additionally, a simple HTML webpage is included to provide an overview of the banking system.

Introduction

The coding part of Java is divided into four modules, each assigned to a student for collaborative development. The first part is Account Management which allows users to create new bank accounts for depositing, withdrawing, or transferring funds. The second part is User Management. It collects some information about the user like user ID, password, email, address, and contact information. Users can sign up or login in to check the user's information details. The third part is Transaction History & Reporting. It records all the transaction histories and allows users to view the transaction history. The last part is Admin Management. It also stores users' information. It can view all accounts, block users, and manage or view transactions.

Web Page for Banking System (HTML)

A simple HTML webpage provides an overview of the system with the following features:

A complementary static webpage demonstrates front-end skills:

1. Home Page: Title, introductory paragraph, and banking-themed image.
2. Account Overview Table: Displays sample accounts with status (Active/Blocked).
3. User Account Section: Explains features like balance checks and sign-up links.
4. Admin Section: Describes administrative controls.

HTML Elements: Proper use of headings, tables, images, links, and background colors.

Real-World Relevance

This project mimics industry-standard practices, preparing students for software development roles by combining backend logic (Java) with front-end presentation (HTML). It reinforces critical skills like teamwork (module division), debugging, and user-centered design.

By completing this project, students demonstrate readiness for advanced programming challenges while adhering to professional coding standards.

Explanation to project

Our program is a simple system that can provide some bank services, such as creating an account, depositing, withdrawing, and transferring accounts.

The program was divided into four parts. When you start it, it will give the user a list of the functions, and you can choose them or exit running with your input. The program was divided into four different parts, and each part had its own type of service.

In the first part, users need to create a new account, and the bank system can help you to deposit or draw your money from your account and transfer your money to another account. These are the basic functions of bank services.

The second part is used to manage personal information, it will record users' personal information, store them, and verify users' identity before they log in to the system. It is an important part of strengthening security. Also, users can check and change their information every time.

The third part is used to store transaction history information, such as transaction type, date, and amount of money. It can also help users to calculate the total deposit or withdrawal amount. Users can enter their account ID, and check all of this information in this program.

The last part is used to manage information of this system, and the normal users are limited to using it. The administrator can check all of the accounts which are stored, and block a user to prevent him from using the system. This function can help the administrator check Illegal users, and find all of the information and transaction history.

Chapter 4

Features and Functionality:

This project combines a Java backend and a static HTML frontend to create a functional banking system. Below are the key features and functionalities organized by module.

1. Account Management

Account Creation: Create new accounts with unique IDs, holder names, and types (e.g., Savings, Checking).

Balance Operations: Deposit funds with validation (e.g., prevent negative amounts). Withdraw funds with overdraft protection (balance cannot go below \$0).

Fund Transfers: Transfer money between accounts using account IDs. Validate source and destination accounts before processing transactions.

2. User Management

User Registration: Register new users with unique IDs, usernames, passwords, and emails. Store users in an `ArrayList<User>` or `HashMap`.

User Authentication: Login with username/password credentials. Basic password validation (e.g., minimum length).

Profile Management: View and update user details (e.g., email, contact information).

3. Transaction History & Reporting

Transaction Logging: Automatically record deposits, withdrawals, and transfers in an `ArrayList<Transaction>`. Track transaction details: type, amount, date, and associated account.

Customizable Reports: Filter transactions by type (deposit/withdrawal/transfer), date range, or amount. Sort transactions chronologically.

Audit Trail: Generate reports for users or admins to review financial activity.

4. Admin Management

Account Oversight: View all accounts, including balances and statuses (Active/Blocked).

User Moderation: Block/unblock users to restrict system access.

Transaction Oversight: Review and audit all transactions across accounts.

Access Control: Admins inherit from the User class but have elevated privileges (e.g., blocking users).

Chapter 4

Challenge and Solutions:

Challenge 1: Understanding Class Structures

Question: What are the main attributes of the Account class? What is their purpose?

Solution: The Account class has the following attributes. AccountID is for storing the unique identifier for the account. AccountHolderName is for storing the name of the account holder. Balance is for storing the amount of money in the account. AccountType is for defining whether the account is savings, checking, etc. Each attribute helps define an account uniquely and is used in various methods like deposit and withdrawal.

Challenge 5: Identifying Possible Errors

Question: What happens if a user enters a username that does not exist during login?

Solution: If the username is not found in the NAME list, the program should output a message like: "User not found. Please try again." If no such check is implemented, the program may cause an IndexOutOfBoundsException when trying to find the password at a non-existent index.

Challenge: Dynamic Table Generation

Question: We don't know how to generate a table dynamically with the input from the user.

Solution: We looked up the tutorial from the html website (<https://www.w3schools.com/html/>) and learned how to do it.

Chapter-5

HTML Implementation:

```
<!DOCTYPE html>
<html>
<body>

<h1 style="background-color:CadetBlue;">Bank Website</h1>
<p>Our bank offers include account creation, loans, deposits.


</body>
</html>

<html>
<body>
<style>
table, th, td {
    border: 1px solid white;
    border-collapse: collapse;
}
th, td {
    background-color: #96D4D4;
}
</style>
</body>
</html>

<html>
<body>
<h2 style="background-color:CadetBlue;">Account Overview</h2>
<form id="accountForm">
    <label for="accountId">Account ID:</label><br>
    <input type="text" id="accountId" required><br>

    <label for="accountName">Account Holder Name:</label><br>
    <input type="text" id="accountName" required><br>

    <label for="balance">Balance:</label><br>
    <input type="number" id="balance" required><br>

    <label for="accountType">Account Type:</label><br>
    <input type="text" id="accountType" required><br>

    <label for="status">Status:</label><br>
    <input type="text" id="status" required><br>

    <button type="button" onclick="addAccount()">Add Account</button>
</form>
```

```

<h3 style="color:CadetBlue;">Account List</h3>
<table id="accountTable">
  <thead>
    <tr>
      <th>Account ID</th>
      <th>Account Holder Name</th>
      <th>Balance</th>
      <th>Account Type</th>
      <th>Status</th>
    </tr>
  </thead>
  <tbody>

  </tbody>
</table>

</body>
</html>

<html>
<body>
<h2 style="background-color:CadetBlue;">User Account Section</h2>
<p>In this page, users can <b>create accounts, check balances, and view transaction history</b>.</p>
<a href="#">Sign Up</a >

</body>
</html>

<html>
<body>
<h2 style="background-color:CadetBlue;">User Account Section</h2>
<p>admins can <b>view, block</b> users, and manage transactions</p>

```

```

<script>
  function addAccount() {

    const accountId = document.getElementById('accountId').value;
    const accountName = document.getElementById('accountName').value;
    const balance = document.getElementById('balance').value;
    const accountType = document.getElementById('accountType').value;
    const status = document.getElementById('status').value;

    const tableBody =
document.getElementById('accountTable').getElementsByTagName('tbody')[0];

    const newRow = tableBody.insertRow();

    newRow.insertCell(0).textContent = accountId;
    newRow.insertCell(1).textContent = accountName;
    newRow.insertCell(2).textContent = balance;
    newRow.insertCell(3).textContent = accountType;
    newRow.insertCell(4).textContent = status;

    document.getElementById('accountForm').reset();
  }
</script>

<style>
  body {
    background-color: AliceBlue;
  }
</style>

</body>
</html>

```

Webpage

<https://bank-mangement.w3spaces.com/saved-from-Tryit-2025-03-05-b88q0.html>

Bank Website

Our bank offers include account creation, loans, deposits.  Trulli

Account Overview

Account ID:

Account Holder Name:

Balance:

Account Type:

Status:

[Add Account](#)

Account List

Account ID	Account Holder Name	Balance	Account Type	Status
123	Mike	1000	savings	Active
111	John	500	checking account	Blocked

User Account Section

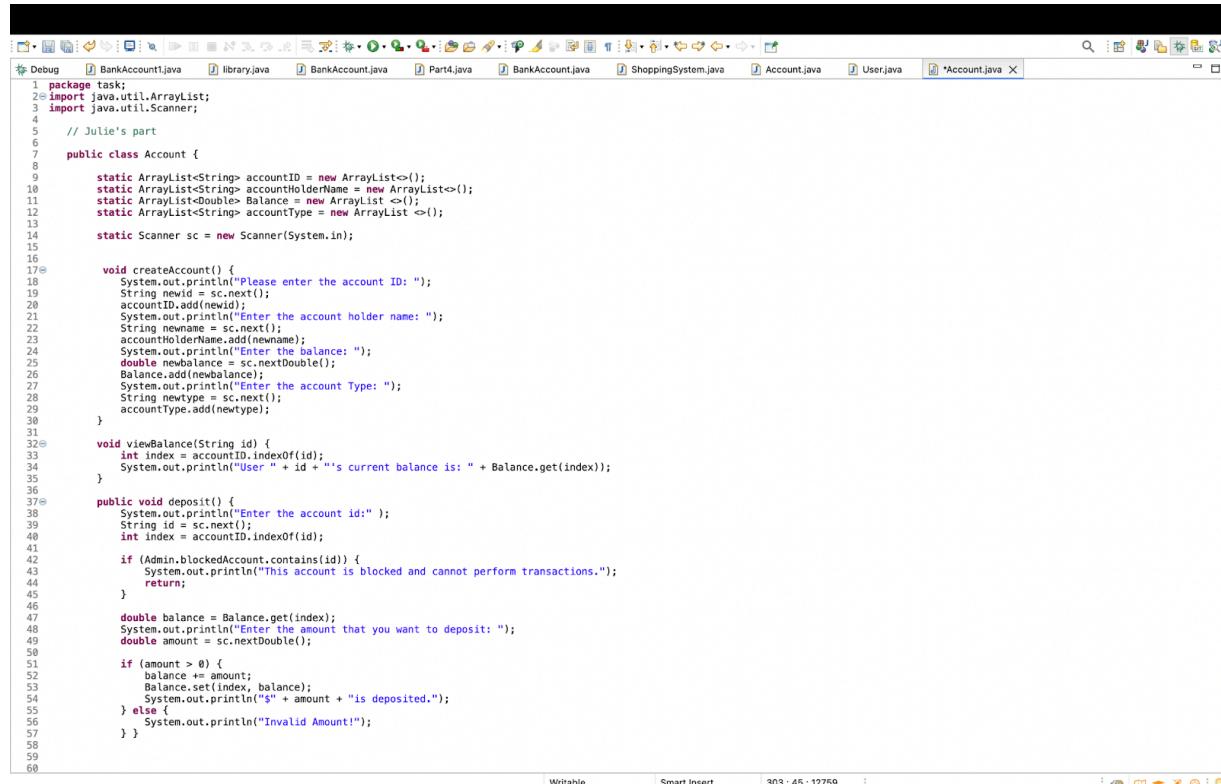
In this page, users can **create accounts, check balances, and view transaction history.**

[Sign Up](#)

User Account Section

admins can **view, block** users, and manage transactions

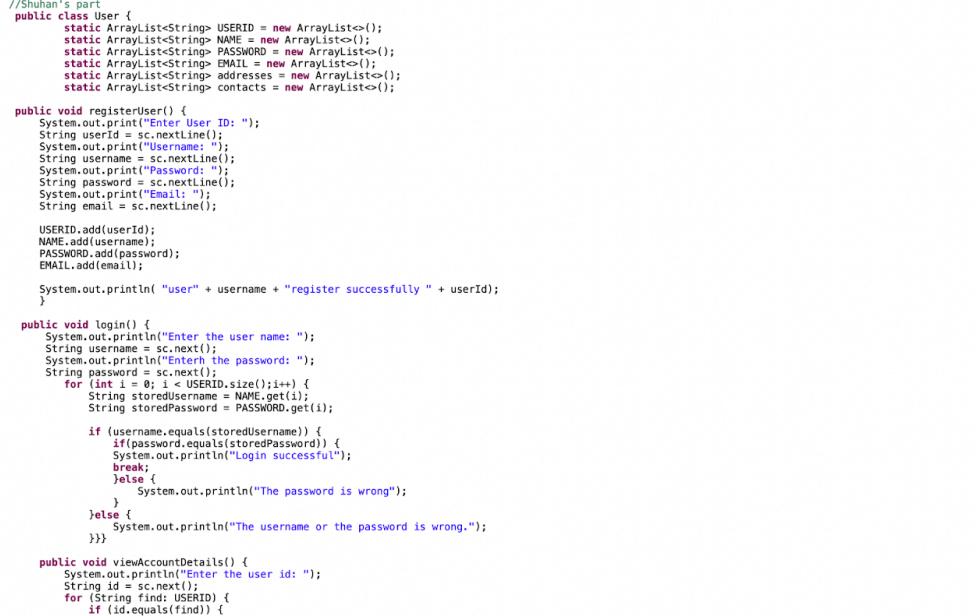
Code:



```

1 package task;
2 import java.util.ArrayList;
3 import java.util.Scanner;
4
5 // Julie's part
6
7 public class Account {
8
9     static ArrayList<String> accountID = new ArrayList<>();
10    static ArrayList<String> accountHolderName = new ArrayList<>();
11    static ArrayList<Double> Balance = new ArrayList<>();
12    static ArrayList<String> accountType = new ArrayList<>();
13
14    static Scanner sc = new Scanner(System.in);
15
16
17    void createAccount() {
18        System.out.print("Please enter the account ID: ");
19        String nextID = sc.nextLine();
20        accountID.add(nextID);
21        System.out.print("Enter the account holder name: ");
22        String newname = sc.nextLine();
23        accountHolderName.add(newname);
24        System.out.print("Enter the balance: ");
25        double newbalance = sc.nextDouble();
26        Balance.add(newbalance);
27        System.out.print("Enter the account Type: ");
28        String newtype = sc.nextLine();
29        accountType.add(newtype);
30    }
31
32    void viewBalance(String id) {
33        int index = accountID.indexOf(id);
34        System.out.println("User " + id + "'s current balance is: " + Balance.get(index));
35    }
36
37    public void deposit() {
38        System.out.print("Enter the account id: ");
39        String id = sc.nextLine();
40        int index = accountID.indexOf(id);
41
42        if (Admin.blockedAccount.contains(id)) {
43            System.out.println("This account is blocked and cannot perform transactions.");
44            return;
45        }
46
47        double balance = Balance.get(index);
48        System.out.print("Enter the amount that you want to deposit: ");
49        double amount = sc.nextDouble();
50
51        if (amount > 0) {
52            balance += amount;
53            Balance.set(index, balance);
54            System.out.println("$" + amount + " is deposited.");
55        } else {
56            System.out.println("Invalid Amount!");
57        }
58    }
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
767
768
769
769
770
771
772
773
774
775
776
777
777
778
779
779
780
781
782
783
784
785
786
787
787
788
789
789
790
791
792
793
794
795
796
797
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
817
818
819
819
820
821
822
823
824
825
826
827
827
828
829
829
830
831
832
833
834
835
836
837
837
838
839
839
840
841
842
843
844
845
846
847
847
848
849
849
850
851
852
853
854
855
856
857
857
858
859
859
860
861
862
863
864
865
866
867
867
868
869
869
870
871
872
873
874
875
876
877
877
878
879
879
880
881
882
883
884
885
886
887
887
888
889
889
890
891
892
893
894
895
896
897
897
898
899
899
900
901
902
903
904
905
906
907
907
908
909
909
910
911
912
913
914
915
916
917
917
918
919
919
920
921
922
923
924
925
926
926
927
928
928
929
930
931
932
933
934
935
936
937
937
938
939
939
940
941
942
943
944
945
946
946
947
948
948
949
950
951
952
953
954
955
956
957
957
958
959
959
960
961
962
963
964
965
966
966
967
968
968
969
969
970
971
972
973
974
975
976
977
977
978
979
979
980
981
982
983
984
985
986
987
987
988
989
989
990
991
992
993
994
995
996
997
998
999
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1026
1027
1028
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1046
1047
1048
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1066
1067
1068
1068
1069
1070
1071
1072
1073
1074
1075
1076
1076
1077
1078
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1087
1088
1089
1089
1090
1091
1092
1093
1094
1095
1096
1096
1097
1098
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1126
1127
1128
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1137
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1146
1147
1148
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1166
1167
1168
1168
1169
1170
1171
1172
1173
1174
1175
1176
1176
1177
1178
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1187
1188
1189
1189
1190
1191
1192
1193
1194
1195
1196
1196
1197
1198
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1216
1217
1218
1218
1219
1220
1221
1222
1223
1224
1225
1226
1226
1227
1228
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1237
1238
1239
1239
1240
1241
1242
1243
1244
1245
1246
1246
1247
1248
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1266
1267
1268
1268
1269
1270
1271
1272
1273
1274
1275
1276
1276
1277
1278
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1287
1288
1289
1289
1290
1291
1292
1293
1294
1295
1296
1296
1297
1298
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1316
1317
1318
1318
1319
1320
1321
1322
1323
1324
1325
1326
1326
1327
1328
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1337
1338
1339
1339
1340
1341
1342
1343
1344
1345
1346
1346
1347
1348
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1357
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1366
1367
1368
1368
1369
1370
1371
1372
1373
1374
1375
1376
1376
1377
1378
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1387
1388
1389
1389
1390
1391
1392
1393
1394
1395
1396
1396
1397
1398
1398
1399
1400
1401
1402
1403
1404
1405
1406
1406
1407
1408
1408
1409
1410
1411
1412
1413
1414
1415
1415
1416
1417
1417
1418
1419
1420
1421
1422
1423
1424
1425
1425
1426
1427
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1436
1437
1438
1438
1439
1440
1441
1442
1443
1444
1445
1446
1446
1447
1448
1448
1449
1450
1451
1452
1453
1454
1455
1456
1456
1457
1458
1458
1459
1460
1461
1462
1463
1464
1465
1466
1466
1467
1468
1468
1469
1470
1471
1472
1473
1474
1475
1476
1476
1477
1478
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1487
1488
1489
1489
1490
1491
1492
1493
1494
1495
1496
1496
1497
1498
1498
1499
1500
1501
1502
1503
1504
1505
1506
1506
1507
1508
1508
1509
1510
1511
1512
1513
1514
1515
1515
1516
1517
1517
1518
1519
1520
1521
1522
1523
1524
1525
1525
1526
1527
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1536
1537
1538
1538
1539
1540
1541
1542
1543
1544
1545
1546
1546
1547
1548
1548
1549
1550
1551
1552
1553
1554
1555
1556
1556
1557
1558
1558
1559
1560
1561
1562
1563
1564
1565
1565
1566
1567
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1576
1577
1578
1578
1579
1580
1581
1582
1583
1584
1585
1586
1586
1587
1588
1588
1589
1590
1591
1592
1593
1594
1595
1595
1596
1597
1597
1598
1599
1599
1600
1601
1602
1603
1604
1605
1605
1606
1607
1607
1608
1609
1609
1610
1611
1612
1613
1614
1615
1615
1616
1617
1617
1618
1619
1620
1621
1622
1623
1624
1625
1625
1626
1627
1627
1628
1629
1630
1631
1632
1633
1634
1635
1635
1636
1637
1637
1638
1639
1640
1641
1642
1643
1644
1644
1645
1646
1646
1647
1648
1648
1649
1650
1651
1652
1653
1654
1655
1655
1656
1657
1657
1658
1659
1660
1661
1662
1663
1664
1665
1665
1666
1667
1667
1668
1669
1670
1671
1672
1673
1674
1675
1675
1676
1677
1677
1678
1679
1680
1681
1682
1683
1684
1685
1685
1686
1687
1687
1688
1689
1689
1690
1691
1692
1693
1694
1695
1695
1696
1697
1697
1698
1699
1699
1700
1701
1702
1703
1704
1705
1705
1706
1707
1707
1708
1709
1709
1710
1711
1712
1713
1714
1715
1715
1716
1717
1717
1718
1719
1720
1721
1722
1723
1724
1725
1725
1726
1727
1727
1728
1729
1730
1731
1732
1733
1734
1735
1735
1736
1737
1737
1738
1739
1740
1741
1742
1743
1744
1745
1745
1746
1747
1747
1748
1749
1750
1751
1752
1753
1754
1755
1755
1756
1757
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1765
1766
1767
1767
1768
1769
1769
1770
1771
1772
1773
1774
1775
1775
1776
1777
1777
1778
1779
1779
1780
1781
1782
1783
1784
1785
1785
1786
1787
1787
1788
1789
1789
1790
1791
1792
1793
1794
1795
1795
1796
1797
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1805
1806
1807
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1815
1816
1817
1817
1818
1819
1820
1821
1822
1823
1824
1825
1825
1826
1827
1827
1828
1829
1829
1830
1831
1832
1833
1834
1835
1835
1836
1837
1837
1838
1839
1839
1840
1841
1842
1843
1844
1845
1845
1846
1847
1847
1848
1849
1849
1850
1851
1852
1853
1854
1855
1855
1856
1857
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1865
1866
1867
1867
1868
1869
1869
1870
1871
1872
1873
1874
1875
1875
1876
1877
1877
1878
1879
1879
1880
1881
1882
1883
1884
1885
1885
1886
1887
1887
1888
1889
1889
1890
1891
1892
1893
1894
1894
1895
1896
1896
1897
1898
1898
1899
1900
1901
1902
1903
1904
1904
1905
1906
1906
1907
1908
1908
1909
1910
1911
1912
1913
1914
1914
1915
1916
1916
1917
1918
1918
1919
1920
1921
1922
1923
1924
1924
1925
1926
1926
1927
1928
1928
1929
1930
1931
1932
1933
1934
1934
1935
1936
1936
1937
1938
1938
1939
1940
1941
1942
1943
1944
1944
1945
1946
1946
1947
1948
1948
1949
1950
1951
1952
1953
1954
1954
1955
1956
1956
1957
1958
1958
1959
1960
1961
1962
1963
1964
1964
1965
1966
1966
1967
1968
1968
1969
1970
1971
1972
1973
1974
1974
1975
1976
1976
1977
1978
1978
1979
1980
1981
1982
1983
1984
1984
1985
1986
1986
1987
1988
1988
1989
1990
1991
1992
1993
1993
1994
1995
1995
1996
1997
1997
1998
1999
1999
2000
2001
2002
2003
2003
2004
2005
2005
2006
2007
2007
2008
2009
2009
2010
2011
2012
2013
2013
2014
2015
2015
2016
2017
2017
2018
2019
2019
2020
2021
2022
2023
2023
2024
2025
2025
2026
2027
2027
2028
2029
2029
2030
2031
2032
2033
2034
2034
2035
2036
2036
2037
2038
2038
2039
2040
2041
2042
2043
2044
2044
2045
2046
2046
2047
2048
2048
2049
2050
2051
2052
2053
2054
2054
2055
2056
2056
2057
2058
2058
2059
2060
2061
2062
2063
2064
2064
2065
2066
2066
2067
2068
2068
2069
2070
2071
2072
2073
2074
2074
2075
2076
2076
2077
2078
2078
2079
2080
2081
2082
2083
2084
2084
2085
2086
2086
2087
2088
2088
2089

```



The screenshot shows an IDE interface with multiple tabs open. The active tab contains Java code for a shopping system. The code includes methods for registering users, logging in, and viewing account details. It uses static ArrayLists to store user information like User ID, Name, Password, Email, Address, and Contacts. The code is annotated with line numbers from 114 to 173.

```
114
115
116 //Shuhan's part
117 public class User {
118     static ArrayList<String> USERID = new ArrayList<>();
119     static ArrayList<String> NAME = new ArrayList<>();
120     static ArrayList<String> PASSWORD = new ArrayList<>();
121     static ArrayList<String> EMAIL = new ArrayList<>();
122     static ArrayList<String> addresses = new ArrayList<>();
123     static ArrayList<String> contacts = new ArrayList<>();
124
125     public void registerUser() {
126         System.out.print("Enter User ID: ");
127         String userId = sc.nextLine();
128         System.out.print("Name: ");
129         String username = sc.nextLine();
130         System.out.print("Password: ");
131         String password = sc.nextLine();
132         System.out.print("Email: ");
133         String email = sc.nextLine();
134
135         USERID.add(userId);
136         NAMES.add(username);
137         PASSWORD.add(password);
138         EMAIL.add(email);
139
140         System.out.println("User " + username + " register successfully " + userId);
141     }
142
143     public void login() {
144         System.out.println("Enter the user name: ");
145         String username = sc.next();
146         System.out.println("Enter the password: ");
147         String password = sc.next();
148         for (int i = 0; i < USERID.size(); i++) {
149             String storedUsername = NAME.get(i);
150             String storedPassword = PASSWORD.get(i);
151
152             if (username.equals(storedUsername)) {
153                 if (password.equals(storedPassword)) {
154                     System.out.println("Login successful");
155                     break;
156                 } else {
157                     System.out.println("The password is wrong");
158                 }
159             } else {
160                 System.out.println("The username or the password is wrong.");
161             }
162         }
163
164         public void viewAccountDetails() {
165             System.out.println("Enter the user id: ");
166             String id = sc.next();
167             for (String find : USERID) {
168                 if (id.equals(find)) {
169                     int index = USERID.indexOf(find);
170                     System.out.println("UserId: " + id);
171                     System.out.println("Username: " + NAME.get(index));
172                     System.out.println("Email: " + EMAIL.get(index));
173                     System.out.println("Address: " + addresses.get(index));
174                     System.out.println("Contact information: " + contacts.get(index));
175                 }
176             }
177         }
178     }
179 }
```



The screenshot shows an IDE interface with multiple tabs open. The tabs include: Debug, BankAccount1.java, library.java, BankAccount.java, Part4.java, BankAccount.java, ShoppingSystem.java, Account.java, User.java, and *Account.java X. The main code editor area displays Java code for a banking system, specifically for managing accounts and transactions. The code includes methods for adding transactions, viewing transaction history, generating reports, and managing admin users.

```
226
227     int transactionId = transactionIds.size() + 1;
228     transactionIds.add(transactionId);
229     accountIds.add(id);
230     transactionTypes.add(transactionType);
231     amounts.add(amount);
232     transactionDates.add(transactionDate);
233
234     System.out.println("Transaction added successfully!");
235     return;
236 }
237
238 System.out.println("Account not found!");
239
240 public void viewTransactionHistory() {
241     System.out.print("Enter your account ID: ");
242     String id = Account.sc.next();
243
244     for (int i = 0; i < accountIds.size(); i++) {
245         if (accountIds.get(i).equals(id)) {
246             System.out.println("Transaction ID: " + transactionIds.get(i));
247             System.out.println("Transaction Type: " + transactionTypes.get(i));
248             System.out.println("Amount: " + amounts.get(i));
249             System.out.println("Date: " + transactionDates.get(i));
250         }
251     }
252 }
253
254 public void generateReport() {
255     System.out.print("Enter your account ID: ");
256     String checkId = Account.sc.next();
257
258     double totalDeposit = 0;
259     double totalWithdraw = 0;
260
261     for (int i = 0; i < accountIds.size(); i++) {
262         if (accountIds.get(i).equals(checkId)) {
263             if (transactionTypes.get(i).equalsIgnoreCase("Deposit")) {
264                 totalDeposit += amounts.get(i);
265             } else if (transactionTypes.get(i).equalsIgnoreCase("Withdrawal")) {
266                 totalWithdraw += amounts.get(i);
267             }
268         }
269     }
270
271     System.out.println("Total Deposits: " + totalDeposit);
272     System.out.println("Total Withdrawals: " + totalWithdraw);
273 }
274
275 // Debbie's part
276 class Admin extends User {
277
278     ArrayList<String> blockedAccount= new ArrayList<>();
279
280     public void viewAllAccounts() {
281         for (int i = 0; i < accountID.size(); i++) {
282             System.out.println(accountID.get(i) + " " + accountHolderName.get(i) + " " + Balance.get(i));
283         }
284
285     public void blockUser() {
286         System.out.println("Please enter the account ID that you want to blocked: ");
287         String Block = sc.next();
288     }
289 }
```

```
174         }
175     }
176     public void updateAccountInfo() {
177         System.out.println("Enter the user ID: ");
178         String id = sc.nextLine();
179         for (int i = 0; i < USERID.size(); i++) {
180             if (id.equals(USERID.get(i))) {
181                 System.out.print("Enter new address: ");
182                 String newAddress = sc.nextLine();
183                 addresses.set(i, newAddress);
184             }
185             System.out.print("Enter new contact info: ");
186             String newContactInfo = sc.nextLine();
187             contacts.set(i, newContactInfo);
188         }
189     }
190 }
191
192 //Sienna's part
193 class Transaction {
194     private int transactionId;
195     private String accountId;
196     private String transactionType;
197     private double total;
198     private String date;
199
200     static ArrayList<Integer> transactionIds = new ArrayList<>();
201     static ArrayList<String> accountIds = new ArrayList<>();
202     static ArrayList<String> transactionTypes = new ArrayList<>();
203     static ArrayList<double> amounts = new ArrayList<>();
204     static ArrayList<String> transactionDates = new ArrayList<>();
205
206     public Transaction(int transactionId, String accountId, String transactionType, double amount, String transactionDate) {
207         this.transactionId = transactionId;
208         this.accountId = accountId;
209         this.transactionType = transactionType;
210         this.total = amount;
211         this.date = transactionDate;
212     }
213
214     public void addTransaction() {
215         System.out.print("Enter Account ID: ");
216         String id = Account.sc.nextLine();
217
218         for (int i = 0; i < Account.accountId.size(); i++) {
219             if (Account.accountId.get(i).equals(id)) {
220                 System.out.print("Enter Transaction Type (Deposit or Withdrawal): ");
221                 String transactionType = sc.nextLine();
222                 System.out.print("Enter Transaction Date (YYYY-MM-DD): ");
223                 String transactionDate = sc.nextLine();
224                 System.out.print("Enter Amount: ");
225                 double amount = sc.nextDouble();
226
227                 int transactionId = transactionIds.size() + 1;
228                 transactionIds.add(transactionId);
229                 accountIds.add(id);
230                 transactionTypes.add(transactionType);
231                 amounts.add(amount);
232                 transactionDates.add(transactionDate);
233             }
234         }
235     }
236 }
```

```
282
283     public void blockUser() {
284         System.out.println("Please enter the account ID that you want to blocked: ");
285         String Block = sc.nextLine();
286         System.out.println("Account" + accountID + " is blocked.");
287     }
288
289     public void viewTransactions() {
290         System.out.print("Enter Account ID to view transactions: ");
291         String accId = sc.nextLine();
292         System.out.println("Transactions for Account ID " + accountID + ":");
293         for (int i = 0; i < Transaction.accountIds.size(); i++) {
294             if (Transaction.accountIds.get(i).equals(accountId)) {
295                 System.out.print(Transaction.transactionIds.get(i) + " " + Transaction.transactionTypes + " $" + Transaction.amounts.get(i) + " " + Transaction.transactionDates);
296             }
297         }
298     }
299     public class Main {
300         public static void main(String[] args) {
301             Scanner sc = new Scanner(System.in);
302
303             Account account = new Account();
304             User user = new User();
305             Admin admin = new Admin();
306             Transaction transaction = new Transaction(0, null, null, 0, null);
307
308             while (true) {
309                 System.out.println("-----MENU-----");
310                 System.out.println("Which service do you want?");
311                 System.out.println("1. Bank Account Management");
312                 System.out.println("2. User Management");
313                 System.out.println("3. Transaction History & Report Management");
314                 System.out.println("4. Admin Management");
315                 System.out.println("Enter your choice: ");
316
317                 int choice = sc.nextInt();
318
319                 if (choice == 1) {
320                     System.out.println("-----Bank Account Management-----");
321                     System.out.println("1. Create a new account");
322                     System.out.println("2. View Balance");
323                     System.out.println("3. Deposit");
324                     System.out.println("4. Withdraw");
325                     System.out.println("5. Transfer funds");
326                     System.out.print("Enter your choice: ");
327
328                     int option = sc.nextInt();
329
330                     if (option == 1) {
331                         account.createAccount();
332                     } else if (option == 2) {
333                         System.out.print("Enter account ID: ");
334                         String id = sc.nextLine();
335
336                         if (account.accountId.equals(id)) {
337                             System.out.println("Your current balance is: " + account.balance);
338                         } else {
339                             System.out.println("Account not found!");
340                         }
341                     }
342
343                     if (option == 3) {
344                         System.out.print("Enter amount to deposit: ");
345                         double deposit = sc.nextDouble();
346                         account.deposit(deposit);
347                     }
348
349                     if (option == 4) {
350                         System.out.print("Enter amount to withdraw: ");
351                         double withdraw = sc.nextDouble();
352                         account.withdraw(withdraw);
353                     }
354
355                     if (option == 5) {
356                         System.out.print("Enter account ID of recipient: ");
357                         String id = sc.nextLine();
358
359                         if (account.accountId.equals(id)) {
360                             System.out.print("Enter amount to transfer: ");
361                             double transfer = sc.nextDouble();
362                             account.transferFunds(transfer);
363                         } else {
364                             System.out.println("Recipient account not found!");
365                         }
366                     }
367
368                 }
369             }
370         }
371     }
372 }
```

```
326         if (option == 1) {
327             account.createAccount();
328         } else if (option == 2) {
329             System.out.print("Enter account ID: ");
330             String id = sc.nextLine();
331             account.viewBalance(id);
332         } else if (option == 3) {
333             account.deposit();
334         } else if (option == 4) {
335             account.withdraw();
336         } else if (option == 5) {
337             account.transferFunds();
338         }
339     } else if (choice == 2) {
340         System.out.println("-----User Management-----");
341         System.out.println("1. Register User");
342         System.out.println("2. Login");
343         System.out.println("3. View Account Details");
344         System.out.println("4. Update Account Info");
345         System.out.print("Enter your choice: ");
346         int option = sc.nextInt();
347
348         if (option == 1) {
349             user.registerUser();
350         } else if (option == 2) {
351             user.login();
352         } else if (option == 3) {
353             user.viewAccountDetails();
354         } else if (option == 4) {
355             user.updateAccountInfo();
356         }
357     } else if (choice == 3) {
358         System.out.println("-----Transaction History & Report Management-----");
359         System.out.println("1. Add Transaction");
360         System.out.println("2. View Transaction History");
361         System.out.println("3. Generate Report");
362         System.out.print("Enter your choice: ");
363         int option = sc.nextInt();
364
365         if (option == 1) {
366             transaction.addTransaction();
367         } else if (option == 2) {
368             transaction.viewTransactionHistory();
369         } else if (option == 3) {
370             transaction.generateReport();
371         }
372     } else if (choice == 4) {
373         System.out.println("-----Admin Management-----");
374         System.out.println("1. View All Accounts");
375         System.out.println("2. Block User");
376         System.out.println("3. View Transactions");
377         System.out.print("Enter your choice: ");
378         int option = sc.nextInt();
379
380         if (option == 1) {
381             admin.viewAllAccounts();
382         } else if (option == 2) {
383             admin.blockUser();
384         } else if (option == 3) {
385             admin.viewTransactions();
386         }
387     }
388 }
389 }
390 }
391 }
392 }
393 }
394 }
```

Chapter-6

Conclusion

In conclusion, the program using Java, HTML, and Eclipse, creates a bank management system with essential banking functions. We use Java to implement terminal logic to handle the management of accounts, users and transactions. HTML used to build front-end web pages, providing user interfaces and decorate with tables, images, links, and background colors. Moreover, Eclipse is used for writing, debugging, and running Java code.

The bank management system core functionalities include:

- Account management: creating accounts, deposit and withdraw funds, view balances and transferring funds.
- User management: registering, logging in, viewing and updating account information.
- Transaction management: recording and viewing transaction history, as well as generating transaction reports.
- Admin management: view all accounts, block users, and manage transactions.

Overall, the program provides an efficient and friendly method to help users and administrators to handle their accounts smoothly.

Chapter-7

Project Allocation

- Coding Part 1 – Julie
 Part 2 – Shuhan
 Part 3 – Mingxi
 Part 4 – Debbie
 All -- Debbie
- HTML Mingxi
- Report Julie(Chapter 2, Chapter 6)
 Shuhan(Format, declaration, abstract, table of content, Chapter 1, Chapter2)
 Debbie(Chapter2, Chapter 3, Chapter 4, Chapter 5)
 Mingxi(Chapter 2, Chapter 3, Chapter 4)
- PPT Shuhan

References