SENG 300 Winter 2020

TEAM MEMBERS

Kaiwen Jia 30050173 jiakaiwen5216@gmail.com
L02 T05 TA: Pavankumar Karkekopp

Haohu Shen 30063099 haohu.shen@ucalgary.ca
L02 T04 TA: Eric Austin

Juwei Wang 30053278 juwei.wang@ucalgary.ca
L02 T02 TA: Pavankumar Karkekopp

Xudong Miao 30050905 xudong.miao1@ucalgary.ca
L02 T09 TA: Lorans Alabood

Yifei Lin 30043746 yifei.lin1@ucalgary.ca
L02 T09 TA: Lorans Alabood

Final Report

By
Group 02 Lecture 02

April 12, 2020

Iteration 1

Product Backlog

Our project is to create a Hotel Online Booking System, here are all the user stories we considered so far for our project.

- As a client, I want to see all of types of rooms which the hotel has on the room reservation website page.
- As a client, I want to know how many rooms whose type I would like are available on the room reservation page.
- As a client, I want to make reservations with my name and phone number through the reservation webpage of the hotel when I do not have a account of the hotel official website.
- As a client, I want to create an account of the hotel official website.
- As a client, I want to upload my personal information(name, nickname, phone number, email and so on) to my account so that I can reserve rooms with this.
- As a client, I want to make reservations with my account on the reservation website page.
- As a client, I want to check my booking(order) on the order page.
- As a client, I want to cancel the booking(order) on the order page.
- As an administrator(hotel staff), I want to log in the dashboard of backend management.
- As an administrator(hotel staff), I want to change the information of the booking(order) for clients.
- As an administrator(hotel staff), I want to help clients to check-in on the client on the management page of the hotel when clients who have reserved come to the hotel.
- As an administrator(hotel staff), I want to help clients to check-out through the management webpage of the hotel if the client want to leave.
- As an administrator(hotel staff), I want to extend stay for rooms of clients through the management webpage of the hotel.
- As an administrator(hotel staff), I want to cancel room reservations for clients through the management webpage of the hotel.
- As an administrator(hotel staff), I want to create room reservations for clients through the management webpage of the hotel.
- As an administrator(hotel staff), I want to check current information (including how many rooms left) of rooms through the management webpage of the hotel.
- As an administrator(hotel staff), I want to update a new types of room along with its information through the management webpage of the hotel.
- As an administrator(hotel staff), I want to change current information of specific rooms through the management webpage of the hotel.

Sprint Planning Meeting

Since our project is to make a web-based Hotel Online Booking System with **SSM** (Spring MVC+Spring+MyBatis). In this sprint we make sure that our **persistence layer** is able to work normally.

- First, Juwei will gather the requirement, put them together and make a use case diagram for us. Also, Juwei and Kaiwen will sort out all the product user stories.
- Second, we initially assign tasks according to the **MVC** standards. Haohu and Kaiwen will firstly design the database and tables in **DBMS** using **DDL**(data description language).
- Thirdly, Haohu's task is to write codes for most of persistence layer (**Model** in **MVC**) by using some knowledge from industry in MyBatis. He will confirm that all APIs that connects the database to all methods in the persistent layer work fine.
- Meanwhile, Kaiwen will help Haohu writing some extra test cases. Xudong and Yifei will record our meeting discussion and then make detailed reports. Kaiwen make all parts (user stories and reports) together and sort out an Iteration Report.
- Finally, Haohu and Kaiwen will check the format of the document and submit it as a PDF, with the code written so far as a zip file.

Sprint Backlog

According to the standard of MVC framework, we firstly achieve all of works on persistence layer. After the design of database, we can set corresponding persistent methods for each user story. Kaiwen and Haohu will participate this part.

- As a client, I want to see all of types of rooms which the hotel has on the room reservation website page. TODO:
  - Set a method which return a result set that contains the information of all the types of rooms.
  - Write test cases for it.

Implementer: Haohu, Kaiwen

- As a client, I want to know how many rooms whose type I would like are available on the room reservation page. TODO:
  - Set a method which finds how many rooms left with the unique room number and return the result.
  - Write test cases for it.

Implementer: Haohu

- As a client, I want to make reservations with my name, my ID as well as my phone number through the reservation webpage of the hotel when I do not have a account of the hotel official website. TODO:
    - Set a method which insert client id, client’s name, client’s phone number and room number that the client books to the booking table.
    - Write test cases for it.

Implementer: Haohu

- As a client, I want to check my booking(information of my order) on the order page. TODO:
    - Set a method which querys the order according to the client’s id and the client’s email and the client’s phone number and returns as a result set.
    - Write test cases for it.

Implementer: Haohu, Kaiwen

- As a client, I want to cancel the booking(information of my order) on the order page. TODO:
    - Set a method which deletes the order with the client’s id and the client’s email and the client’s phone number.
    - Write test cases for it.

Implementer: Haohu

- As an administrator(hotel staff), I want to log in the dashboard of backend management. TODO:
    - Set a method that queries if the username and the password the user entered are valid and matched and returns as a result set.
    - Write test cases for it.

Implementer: Haohu

- As an administrator, I want to change the information of the booking(information of my order) for clients. TODO:
    - Set method which updates new information to the order with the client’s id and the client’s email and the client’s phone number.
    - Write test cases for it.

Implementer: Haohu

- As an administrator, I want to help clients to check in on the front desk on the management page of the hotel when clients who have reserved come to the hotel. TODO:
    - Set method which adds record contains client information and roomID into check-in table.
    - Write test cases for it.

Implementer: Haohu

- As an administrator, I want to help clients to check-out through the management webpage of the hotel if the client want to leave. TODO:
    - Set method which deletes record contains client information and roomID into check-in table
    - Write test cases for it.

Implementer: Haohu

- As an administrator, I want to extend the days to stay for rooms of clients through the management webpage of the hotel. TODO:
    - Set method which update new checkout date for the specific record.
    - Write test cases for it.

Implementer: Haohu

- As an administrator, I want to cancel room reservations for clients through the management webpage of the hotel. TODO:
    - Set method which delete the record with specific room number
    - Write test cases for it.

Implementer: Haohu

- As an administrator, I want to create room reservations for clients through the management webpage of the hotel. TODO:
    - The method is same as the method in user story 3.
    - Write test cases for it.

Implementer: Haohu

- As an administrator, I want to check current information (including how many rooms left) of rooms through the management webpage of the hotel. TODO:
    - The method is same as the method in user story 1.
    - Write test cases for it.

Implementer: Haohu

- As an administrator, I want to update a new types of room along with its information through the management webpage of the hotel. TODO:
    - Set method which add new record to the room table return the result.
    - Write test cases for it.

Implementer: Haohu

- As an administrator, I want to change current information of specific rooms through the management webpage of the hotel. TODO:
    - Set method which update new information for a specific room and return the result.
    - Write test cases for it.

Implementer: Haohu

Daily Scrum Meeting

We had 2 scrum meetings for the Iteration 1 in total.

- The first meeting was on Friday, Feb 21th (Juwei Wang, Haohu Shen, Kaiwen Jia, Xudong Miao and Yifei Lin presented the meeting).
    - The meeting was held in the computer science lab. We initially got to know each other by talking about everyone's strength and the part of the project interested in. In this first meeting, as we just got started, we haven't got any progress on the project yet, but we elaborated on how this project is going to be like and we prepared for the next meeting.
    - We also assigned tasks:
        - Yifei and Xudong were going to finish the report.
        - Haohu wanted to do coding part.
        - Juwei has some experience in making projects, so he helped us to gather requirements and make the use case diagram.
        - As the group leader, Kaiwen sorted out the user stories and showed them to Xudong and Yifei so that they can write into Iteration1 as the product backlog.
        - Also, Kaiwen should help Haohu to finish the part of design of database and persistence layer methods and give some details to Xudong and Yifei in order to finish reports.
        - Moreover, we did some configuration for the development environment (IDEA, MAVEN and how to use them).

- This meeting took us about 15 minutes in total.
    - The second scrum meeting was on Sunday Feb 29th. Everyone finished the first draft of the part they take charge of:
        - Xudong and Yifei had finished the outline of the Iteration1 report but still needed some details from other group members.
        - Juwei had finished the use case diagram, and then Kaiwen and Haohu gave some feedback to him.
        - Haohu had ensured the execution of the persistence layer. Kaiwen had written some test files for the persistence layer with Junit and showed that these persistent methods can be used by the business layer.
        - Yifei and Xudong did some notes for these, and they also want to be responsible for the front

end of displaying the page of the hotel system in the next sprint.
    - Kaiwen will do most of the work on the business layer.
    - Haohu will be the free man between frontend and backend to help both if one side needs extra work or looks for encountering a difficulty as Haohu has the most experience in programming among us. This meeting took us about 30 minutes in total.

Sprint Review Meeting

- We hold the first scrum meeting on Friday afternoon Feb 21st, it lasts for 15 mins. All team members attend the meeting.

    - We just tried to understand what should be included in the first sprint and assigned the tasks.
    - Furthermore, we took a lot at some current working products related to the topics of our project. The products include some mature hotel booking websites such as Booking and Expedia, and we spent some hours mainly using the booking and Expedia as a user to see how they work on booking a hotel. Booking and Expedia are both highly rated products and they built a path for us to keep working toward the right direction by using them as a reference. That helps a lot for making a plan for the whole sprint.
    - After the analyzing and evaluating in the first scrum meeting, it gave us some ideas of what our project is going to be like. We firstly clarified our product backlog (Juwei's part), and our main target is to show the user's hotel information and successfully book the hotel for them.
    - Regarding the work we need, there must be a page including the text and pictures related to the information of each shown to the user and we need to store this information in a database. Haohu had tried the best to finish this. Kaiwen had checked all codes and helped Haohu to make test files. Yifei and Xudong finished the whole iteration1 report. In the second scrum meeting, everyone is satisfied with the prod of others, and happy to make great efforts.

Retrospective Meeting

- We held our retrospective meeting on March 5th after the lecture and everyone attended the meeting.
    - This meeting lasted for only 15 minutes as the content required for a retrospective meeting is not much detailed.
    - We have ensured that everything in (Model) persistence layer can normally work. Some expectations are: in the second sprint, we will start working on a business layer and frontend (View level).
    - Filter is planned to add in our system to help the users find the most suitable hotel for them and it can include the location of the hotel, the room type, the additional services and the price range. Other than the things we stick with, we try to avoid any complicated steps which can create inconvenience for the user. We don't demand the users to log in the system for booking the hotel at the first sprint, but they can check in as a guest instead by just filling their basic personal information, and we will

achieve the login function in the second sprint (deal with username and password parameter from the frontend in the business layer).
  - Moreover, in the aspect of creativity, we were also trying something new, such as a rating system for the service of hotel stuff. These can be easily added after we finish the basic frame of our project if we have enough extra time.

Iteration 2

Updated Product Backlog

Our main product backlog is listed in Iteration 1 report. In this Iteration 2 report, we are going to update the newest state(status) of a part user story.

- Story Update 1: As a client, I want to know how many rooms I want left (available) on the reservation page in a period of time(from check-in time to estimate check-out time).

  What we should do is about displaying room information(Room type, amount of left rooms, current price) on the booking page. In Iteration 2 our software should let people choose the time interval of booking.

- Story Update 2: As a client, I want to make reservations with some personal information on the reservation website page.

  This is about the information getting from the front end, we initially made a simple one to send what rooms clients want to book, and the webpage uses js and ajax to produce a JSON style form to the back end.

Updated Sprint Planning Meeting

We had a planning meeting on Mar 15th, Sunday. Because COVID-19 has been spreading, we have to use zoom to have this meeting online. All of the members took part in it. On that day, most of the members told that they had finished learning how to make front end web pages. This week, they believed the work of making login and booking pages can be done. Haohu and Kaiwen plan to implement back-end data parsing and data return services. After this week, maybe we can initially achieve the interactions between front and back ends. Then probably clients can enjoy a complete series of features.

This meeting took 1.5 hours for all the things, and it includes the teaching of development kits(IDE and so on), what kind of data the back end should send and get, and how to use frameworks such as Vue and Vue.js.

Moreover, we should go on to construct the service layer. All of the service interfaces should be implemented. In the next iteration, we will launch the login function. Clients can sign up and book rooms with their account. That is mainly what we want.

Updated Sprint Backlog

- On the booking pages, clients should be able to see how many rooms are left based on a period of dates from check-in to check out. A little bit difficult step is how to return the "amount of left rooms". It relates to how to arrange remaining rooms. We should be able to deal with some problems about clients' length of time to stay, and most of the time these periods are overlapping. However, we should consider these when we calculate the amount of left rooms in the hotel. Therefore, in this iteration, we tried to return these data more accurately. Through applying some methods, the hotel management system can do most of the work on time allocation based on the check-in and check-out date(from users), and give out the amount of rooms left (for just a single type).

- On the booking pages, clients should also be able to submit orders of booking. That means the webpages can produce HTML forms containing information such as client personal info(phone number and names), estimate check-in date and check-out date and the room type. That we have finished so far. Furthermore, We chose 1 member and let him simulate a client. After trying, he is satisfied, but thinks that the webpage should be improved and beautified, thereby being easier to understand and use. So, in the next iteration, we could consider using some technologies like some front end framework BOOTSTRAP.

Daily Scrum Meeting

We had 2 scrum meetings for the Iteration 2 in total.

- The first meeting was on Friday, March 3rd (Juwei Wang, Haohu Shen, Kaiwen Jia, Xudong Miao and Yifei Lin presented the meeting).

    - The meeting was held in the computer science lab. Since we now come to the stage of the second iteration, we make sure every team member sets up the same environment in their devices. Though we may use different OS platforms, we make sure the toolchains and encoding setting for our project are the same.

    - During the meeting, we:

        - make sure every member is developing their part of project by making their personal branch
        - make sure every member can merge other branches properly instead of directly switching the branch.
        - make sure every member understands the relationship between different table and the meaning of each field of every table of our database.
        - decide to change some improper data fields and remove some redundant SQL statements for our database.
        - ask all team members to learn some basic knowledge about Vue.js in order to prepare for the front-end coding.

        - decide not to consider how to avoid DDOS since it may require us to learn tools like Redis and it is a bit of overhead.
        - allocate the job about front-end coding for all team members.
        - This meeting took us about 15 minutes in total.

- The second scrum meeting was on Sunday March 17th. Since the COVID-19, we decide to use Zoom for code-review and remote meetings, during this meeting:

    - Haohu revised and fixed some coding problems in interfaces and Kaiwen gave some feedback to him.

    - We again make sure every member knows the basic login of the business for our project.

    - This meeting took us about 15 minutes in total.

Sprint Review Meeting

We hold 1 sprint review meeting. All group members attended the meeting. The meeting lasts for 1 hour and 15 mins. We reviewed our product and decided to make some changes.

- Our product is a software to help customers book the hotel. Compared to some professional hotel booking websites(for example: booking.com), we found that we need to show more information to the customers.

- We decided to add more information to help the customer choose the room by staying period (check-in time and check-out time) and also show the number of rooms for different room types.

- One of the team members acts as the customer and said we need to improve the quality of the webpage by making it more beautiful.The information of the customer will be saved in a database system, it includes the basic information to identify customers.

Retrospective Meeting

We hold one retrospective meeting. All group members presented, meeting lasts for 20 mins, in the meeting we make sure that every member should

- keep learning the Spring, Javascript and Vue.js.
- keep working on the interface.
- learn how to integrate all these frameworks(including Git) to IntelliJ IDE
- learn how to write and manipulate SQL statements using Datagrip.

Also, we

- discuss how to try to make the webpage more beautiful, as well as consider using the third-party UI frameworks.
- allow the database stores the identify information of the customer from the web page in the future.
- decide not to hold face-to-face meetings any more due to COVID-19 and replace it with Zoom meetings.

Conclusion and plan for next iteration

- In conclusion, in this period of time, we have not actually added many new features for our project but we discussed a lot of difficulties we encountered during implementing the functions and we also argued what we should do at the next step. Currently, we are facing some challenges like implementing the interface of some of the functions (left rooms returning).

- For the next iteration, we'll consider fulfilling most of the requirements that our clients are asking for, including price system, front end interface, and submitting order information.

Additional Information

An updated list user stories

- As an administrator(a hotel staff), I hope only the client who has registered accounts in the hotel's webpage can book rooms, because hackers can easily fill in trash information and book all available rooms if we don't filter the information. Thus, we **CHANGED** the third user story from the iteration 1, that is, as a client, he or she must register an account to book a room, and phone number/name will become a part of information of this account.

- As the financial adviser, I want to control the limitation of the room that a client can book, more specifically, I hope a client can only book a room, since if someone books too many rooms but cancel some of booking when checking in, it will be a waste of the resources and we may lose money because of it.

Test suites

How did we perform testing activities and what testing techniques we have followed?

- Since we used **myBatis** to write SQL statements and connect between the database and our codebase (except we initialize our tables and some dummy data in a mySQL console directly), we firstly instantiated each table in mySQL as a **POJO** (Plain Ordinary Java Object) class in Java, and then writing the **Mapper** files (where and SQL interfaces at) for each **POJO**, finally we used **Junit** to make unit tests for each **POJO**.

Then in order to test SQL statements, we use **selective testing** and we use the strategy of **Equivalence-based Selection** on it. Since we have at least a set of **CRUD** (Create, read, update and delete) SQL statements for each **POJO**, we write blackbox test cases for each statement in each POJO. Some statements, such as querying items, we also derived a single **SELECT** statements to many with custom requirements (using the keyword **WHERE** to filter) to make sure that the front-end programmers can have more flexibility on it. Some SQL statements are tested but not used in the final stage because we tried to write as many statements as possible to for all possible situations before the design of the front-end part.

We use the entity type **Client** as an example to show the steps we took:

1. We first define its attributes in mySQL, as follow:

```
create table client
(
    id            varchar(36)  not null, /* Use UUID */
    name          varchar(30)  not null,
    gender        char         null,
    phone_number  varchar(20)  null,
    address       varchar(256) null,
    email_address varchar(256) null,
    constraint client_id_uindex
        unique (id)
);
alter table client
    add primary key (id);
```

1. Then we convert it to POJO as follow, notice that we do not write getters, setters or constructors here since all of these will be implemented automatically by the plugin we use:

```
public class Client {
    private String id;  // Use UUID here
    private String name;
    private Character gender;
    private String phoneNumber;
    private String address;
    private String emailAddress;
    private String password;
}
```

1. In order to apply myBatis, we wrote interfaces for each SQL statement and implementations for these interfaces in two Java files, the method signatures are listed below:

```
// Retrieve
List<Client> getClientList();
List<Client> getClientListLimit(Map<String, Object> map);
Client getClientById(String id);
List<Client> getClientByName(String name);
List<Client> getClientByGender(String gender);
List<Client> getClientByPhoneNumber(String phoneNumber);
List<Client> getClientByAddress(String address);
List<Client> getClientByEmailAddress(String emailAddress);

// Create
void insertAClient(Map<String, Object> map);

// Update
void updateClient(Map<String, Object> map);

// Delete
void deleteClientById(String clientId);
```

1. Now, we wrote SQL statements for each method in a XML file, for example, to query all clients and return as a list, we can write it as below:

```
<resultMap id="CheckinMap" type="Checkin">
    <result property="checkinId" column="checkin_id"/>
    <result property="roomNumber" column="room_number"/>
    <result property="clientId" column="client_id"/>
    <result property="checkinDate" column="checkin_date"/>
    <result property="checkoutDate" column="checkout_date"/>
</resultMap>

<select id="getCheckinList" resultMap="CheckinMap">
    select * from checkin;
</select>
```

1. Finally, we use log4j and Junit to write the corresponding unit test, and we check if there is any exception happens or if the result conforms that in mySQL, as below:

```
public void testGetClientList() {
    try {
        List<Client> clientList = clientMapper.getClientList();
        for (Client client : clientList) {
            System.out.println(client);
        }
        logger.info("testGetAllClients test OK!");
    } catch (Exception e) {
        logger.error("testGetAllClients test FAILED!");
        e.printStackTrace();
    }
}
```

- The result of each test case in JUnit will be printed to standard output using **Apache log4j** and will be also saved in an external storage since it will bring us much convenient when we need to trace back.
- We separated all checking functions from mySQL, for example, when we insert a new item of the client, we can check the format of the client's phone number given in mySQL by using constraints with a regular expression, but for better conformity, we write all checking functions in **business logic layer**.
- To test the correctness of the APIs that we use in the front-end part to connect the back-end part, we implemented a logger which will print all information we need to the console and we will check the output line by line.
- On the other hand, under **SpringMVC**, we use **Java Exceptions** to handle different return value from **Tomcat** in **Controller Layer**.
    - For example, in the directory *./src/main/java/ca/ucalgary/seng300/controller/exceptions* of our project, all these class files are sub-classes of *AdminController*:

```
AdminControllerAddAdminEx.java
AdminControllerUpdateNewPasswdEx.java
AdminControllerDeleteUserEx.java
AdminControllerException.java
AdminControllerLoginEx.java
AdminControllerSignInEx.java
AdminControllerUpdateNewNameEx.java
AdminControllerUserNotExistEx.java
```

where each file contains an entity that handles the exceptions for a feature of administrators on the webpage. Also, in each file, such as *AdminControllerLoginEx.java*, we use multiple constructors to handle one exception in different ways, which brings us much flexibility:

```
public AdminControllerLoginEx() {
    super();
}
public AdminControllerLoginEx(String message) {
    super(message);
}
public AdminControllerLoginEx(String message, Throwable cause) {
    super(message, cause);
}
public AdminControllerLoginEx(Throwable cause) {
    super(cause);
}
protected AdminControllerLoginEx(String message, Throwable cause,
boolean enableSuppression, boolean writableStackTrace) {
    super(message, cause, enableSuppression, writableStackTrace);
}
```

- Before adding further components such as CSS, we will retrieve the data coming from back-end and show it using **JSON**, we won't go to next step if the contents of **JSON** is not the same as we expect.

- We also test the correctness of these APIs by giving some invalid data deliberately in our **Black Box Testing** and check if our code can respond correctly. For example, in the page of room booking, we try to use different pairs of invalid (checkin-date, checkout-date) in our inputbox, such as

    - The checkout-date is earlier than the checkin-date, such as (2020-04-10, 2020-04-01).
    - The checkin-date/checkout-date contains digits, but is not a date with the correct format, such as (0410-2020, 2020~04~12).
    - The checkin-date/checkout-date contains other characters than digits, such as (2020-04-i0, 2020-04-o1).

- To test the correctness of the logic of CSS in our webpages, we just simply check if there is any error in our design, such as checking if the input box or other graphic components cannot be correctly resized or refreshed.