Second Iteration

Team Name: Roller Coaster

Team Members:
Yuzhang Yuan yy2979, Yumiao Li yl4225, Guanjie Wang gw2402,
Yuanmeng Xia yx2548, Xinyue Wang xw2647

Part I

MVP User Stories:

1. As a visitor, I can register my own account via my personal information and since then I can log in my account via my phone number and my password when I need to use this application. Besides, I am able to login in my account via Google. My conditions of satisfaction are -
    a. When I register my account, I should provide information including telephone number, name, gender, email, age and password.
    b. When I login my account, I need to input the correct telephone number and corresponding correct password.
    c. If I don't want to register an account, I could also login this application via Google verification.

2. As a visitor, I want to make my reservations and orders and observe them easily and modify them if I need so that I can book reservations and orders online and cancel them if I change my mind. My conditions of satisfaction are-
    a. I can cancel my orders and appointments if I want.
    b. All my orders and appointments will display according to the time.
    c. Users can make appointments and orders, and they must be consistent.
    d. If the server crashes and restarts, users should be able to observe their orders and appointments which have been made.
    e. I can see all the announcements on the website.
    f. When I choose a date, I can see the weather forecast.
    g. The ticket information will be delivered to my email address .

3. As a visitor currently in the park, I want to know the facilities' and events' current conditions so that I can make my in-park plan. My conditions of satisfaction are-
    a. The time when the events are available.
    b. Some description of events
    c. Some introductions about facilities

    d. The time when the facilities are available
    e. How many people are in the queue now.
    f. The new announcement from Roller Coaster amusement park will be delivered to my email address.
4. As a visitor in the park, I want to be able to post a comment or rate the facilities in the park. My conditions of satisfaction are-
    a. I can rate facilities.
    b. I can post a comment about my evaluation to the amusement park.
5. As a visitor, I want to use Quick Pass to appoint facilities in certain time slots. I can view my appointments and delete them if I change my mind. My conditions of satisfaction are-
    a. I can use my QuickPass to appoint facilities.
    b. I can cancel my facility appointments.
    c. I can view all my facility appointments.
6. As a visitor, I want to manage my balance account. My conditions of satisfaction are-
    a. I can charge my account.
    b. I can use the balance in my account to buy tickets.
    c. I can monitor the balance in my account.
7. As a park manager, I want to have permission to change the status of events, add/delete events, check the current reservations for all visitors, and tickets' information so that I can manage the amusement park. My conditions of satisfaction are-
    a. I can add/delete events and facilities.
    b. I can modify the start and end time of events.
    c. I can modify the open and close time of facilities.
    d. I can modify the description of events and introduction of facilities.
    e. I can modify the current queue status which stands for how many people are waiting in the queue of the given facility.
    f. For a certain event, I can modify the number of remaining positions of the given event.
    g. For a certain ticket, I can check whether the ticket is valid or not.
    h. For a certain appointment, I can check whether the appointment is valid or not.
    i. I can add announcements and send notifications to visitors' email.
    j. I can delete announcements.
    k. I can change the ticket price based on the current condition.
    l. I can obtain the statistics of the amusement park such as the top 5 rating facilities and how many people have come to the amusement park on the given date and so on.
    m. For a certain quickPass, I can check whether the quickPass is valid or not.

Part II

equivalence partitions & boundary conditions

User Service:

1. For the register method, we have one valid input and two invalid inputs.
    a. Using the valid input including user personal information about name, gender, role(visitor or manager), telephone number, password and email, the user can register a new account successfully. The test case name is registerWithValidInputTest().
    b. One of the invalid inputs is a null user model when the user information parameters are invalid. Under this condition, the user cannot register an account successfully and the application will return an parameters invalidation error. The test case name is nullUserModelRegister().
    c. If a user registers an account with a duplicate telephone number which has been used by other users, the input user model here is invalid. In this situation the user cannot register an account successfully and the application will throw an exception. The test case name is duplicateRegisterTest().
2. For the validateLogin method using telephone number and password, we have one invalid input and two invalid inputs.
    a. Users can login in via telephone number and corresponding correct password so the valid input is a user model including properties of telephone number and correct password. The test case name is validateLoginTest().
    b. If a user never registered an account, he/ she cannot login using a telephone number and password which are not stored in the database. The invalid input here is a user model whose information does not exist in the database. The test case name is userNotRegisterLoginTest().
    c. If a user input a wrong password(invalid input), he/ she cannot login successfully. The test case name is wrongPasswordLoginTest().

3. For loginWithGoogle method, we have one valid input and one invalid input.
    a. If we use Google Login Api to verify the identity of users successfully, we get the valid input including user's profile including email address, name and ID. Hence, users can log in successfully. The test case name is validGoogleLoginTest().
    b. If the user's identity is not verified by Google, we cannot get a valid ID. Hence, the user will fail to log in. The name of the test case is googleLoginVerificationFailedTest().

Comment Service:

1. For the addComment method used to post comments, we have one valid input and two invalid inputs.

    a. If a visitor can post a comment on the amusement park when he/ she login his/ her account. The name of the test case is addCommentTest().
    b. If a visitor doesn't log in his account, the input null user model is invalid. Then, the visitor cannot post a comment successfully and the application will return an error code. The name of the test cast is notLoginPostCommentTest().
    c. If the user fails to post a comment, the valid input is null comment model. Then, the visitor cannot post a comment successfully and the application will return an error code. The name of the test cast is invalidCommentTest().
2. For the deleteComment method used to delete a comment. Because a user can only delete his own comment, we have one valid input and one invald input.
    a. If a user wants to delete his own comment, the input is valid. Then, he can delete the comment successfully. The test case name is deleteCommentTest().
    b. If a user wants to delete others' comments, the input is invalid. Then, he cannot delete the comment successfully. The test case name is invalidDeletionTest().
3. For the showAllComments method used to show all the comments from visitors, it has no input parameter. Therefore, there is no invalid input and no exceptions except for SQL internal exceptions. The test case name is showCommentTest() and showRecordsTest().

Manager Service:

1. For the changeTicketPrice method, we have one valid input and two invalid inputs.
    a. For the input of the ticket without ticketType, if the manager can find that the input is invalid, the test is passed and vice versa. And the corresponding test case name is changeTicketPriceFail().
    b. For the input of the ticket without price, if the manager can find that the input is invalid, the test is passed and vice versa. And the corresponding test case name is changeTicketPriceFail2().
    c. For the normal ticket, if the manager can find that the input is valid and change the price of that type of ticket, the test is passed and vice versa. And the corresponding test case name is changeTicketPrice().
2. For the addFacility method, we have one valid input and two invalid inputs.
    a. For the input of facility id which has existed in the database, if the manager can find that the id is duplicated, the test is passed and vice versa. And the corresponding test case name is addFacilityWithDuplicateName().
    b. For the input of a facility with an empty id, if the manager can find that the id is empty and the input is invalid, the test is passed and vice versa. And the corresponding test case name is addFacilityWithEmptyName().

    c. For the normal input, if the manager can find that the input is valid and add the facility successfully, the test is passed and vice versa. And the corresponding test case name is addFacility().
3. For the UpdateFacility method, we have one valid input and two invalid inputs.
    a. For the input of facility id which does not exist in the database, if the manager can find that the input is invalid, the test is passed and vice versa. And the corresponding test case name is updateFacilityWithWrongName().
    b. For the input of a facility with an empty id, if the manager can find that the id is empty and the input is invalid, the test is passed and vice versa. And the corresponding test case name is updateFacilityWithEmptyName().
    c. For the normal input, if the manager can find that the input is valid and update the facility successfully, the test is passed and vice versa. And the corresponding test case name is updateFacility().
4. For the DeleteFacility method, we have one valid input and two invalid inputs.
    a. For the input of facility id which does not exist in the database, if the manager can find that the input is invalid, the test is passed and vice versa. And the corresponding test case name is deleteFacilityWithWrongName().
    b. For the input of a facility with an empty id, if the manager can find that the id is empty and the input is invalid, the test is passed and vice versa. And the corresponding test case name is deleteFacilityWithEmptyName().
    c. For the normal input, if the manager can find that the input is valid and delete the facility successfully, the test is passed and vice versa. And the corresponding test case name is deleteFacility().
5. For the addEvent method, we have one valid input and two invalid inputs.
    a. For the input of Event id which has existed in the database, if the manager can find that the id is duplicated, the test is passed and vice versa. And the corresponding test case name is addEventWithDuplicateName().
    b. For the input of an Event with an empty id, if the manager can find that the id is empty and the input is invalid, the test is passed and vice versa. And the corresponding test case name is addEventWithEmptyName().
    c. For the normal input, if the manager can find that the input is valid and add the Event successfully, the test is passed and vice versa. And the corresponding test case name is addEvent().
6. For the UpdateEvent method, we have one valid input and two invalid inputs.
    a. For the input of Event id which does not exist in the database, if the manager can find that the input is invalid, the test is passed and vice versa. And the corresponding test case name is updateEventWithWrongName().
    b. For the input of an Event with an empty id, if the manager can find that the id is empty and the input is invalid, the test is passed and vice versa. And the corresponding test case name is updateEventWithEmptyName().

c. For the normal input, if the manager can find that the input is valid and update the Event successfully, the test is passed and vice versa. And the corresponding test case name is updateEvent().

7. For the DeleteEvent method, we have one valid input and two invalid inputs.
    a. For the input of Event id which does not exist in the database, if the manager can find that the input is invalid, the test is passed and vice versa. And the corresponding test case name is deleteEventWithWrongName().
    b. For the input of an Event with an empty id, if the manager can find that the id is empty and the input is invalid, the test is passed and vice versa. And the corresponding test case name is deleteEventWithEmptyName().
    c. For the normal input, if the manager can find that the input is valid and delete the Event successfully, the test is passed and vice versa. And the corresponding test case name is deleteEvent().
8. For the checkTicket method, we have one valid input, and four invalid inputs.
    a. For the ticket id which does not exist in the database, if the manager can find that the ticket id is wrong then the test is passed and vice versa. And the test case name is checkTicketWithWrongTicketID().
    b. For the ticket which has been used, if the manager can find that the ticket has been used then the test is passed and vice versa. And the test case name is checkTicketWithUsedTicket().
    c. Consider the valid date of the ticket, we have the boundary condition that only if the valid date of the ticket is the same as the current date, the ticket can be checked successfully. Therefore, we have tested the tickets with current date -1, current date and current date + 1 to test our method and the corresponding test case names are checkTicketWithInvalidTicketLastDay(), checkTicketNormal() and checkTicketWithInvalidTicketNextDay().

Announcement Service:

1. For the pushAnnouncement method, we have one valid input and one invalid input.
    a. For announcement with empty text, if the manager can find that the announcement is invalid, the test is passed and vice versa. And the corresponding test case name is testPushAnnouncementFail().
    b. For announcements with normal input, if the manager can find that the announcement is valid and send emails to all the visitors, the test is passed and vice versa. And the corresponding test case name is testPushAnnouncement().
2. For the deleteAnnouncement method, we have one valid input and two invalid input.
    d. For the input of null as the announcement id , if the manager can find that the announcement id is invalid, the test is passed and vice versa. And the corresponding test case name is testDeleteAnnouncementFail1().

e. For the input of the announcement id which does not exist in the database, if the manager can find that the announcement id is invalid, the test is passed and vice versa. And the corresponding test case name is testDeleteAnnouncementFail2().
f. For announcements with normal input, if the manager can find that the announcement is valid and send emails to all the visitors, the test is passed and vice versa. And the corresponding test case name is testDeleteAnnouncement().

Statistic Collection Service:

1. For the PeopleInThatDay method, we have one valid input and four invalid inputs.
    g. For the input of a query without specifying the year, if the manager can find that the input is invalid, the test is passed and vice versa. And the corresponding test case name is testPeopleInThatDayFail().
    h. For the input of a query without specifying the month, if the manager can find that the input is invalid, the test is passed and vice versa. And the corresponding test case name is testPeopleInThatDayFail2().
    i. For the input of a query without specifying the date, if the manager can find that the input is invalid, the test is passed and vice versa. And the corresponding test case name is testPeopleInThatDayFail3().
    j. Considering the input date of the query, we have the boundary condition that only if the date of the query is before the current day, we can obtain the statistic of how many visitors came to the park on that day. Therefore, we have tested the input with current date -1, current date to test our method and the corresponding test case names are testPeopleInThatDayNormal(), testPeopleInThatDayFail4().
2. For the PeopleInThatMonth method, we have one valid input and three invalid inputs.
    a. For the input of a query without specifying the year, if the manager can find that the input is invalid, the test is passed and vice versa. And the corresponding test case name is testPeopleInThatMonthFail().
    b. For the input of a query without specifying the month, if the manager can find that the input is invalid, the test is passed and vice versa. And the corresponding test case name is testPeopleInThatMonthFail2().
    c. Considering the input date of the query, we have the boundary condition that only if the date of the query is before the current month, we can obtain the statistic of how many visitors came to the park in that month. Therefore, we have tested the input with current month-1 and current month to test and the corresponding test case names are testPeopleInThatMonth(), testPeopleInThatMonthFail3().
3. For the PeopleInThatYear method, we have one valid input and two invalid inputs.
    a. For the input of a query without specifying the year, if the manager can find that the input is invalid, the test is passed and vice versa. And the corresponding test case name is testPeopleInThatYearFail().

b. Considering the input date of the query, we have the boundary condition that only if the date of the query is before the current year, we can obtain the statistic of how many visitors came to the park in that year. Therefore, we have tested the input with current year-1 and current year to test and the corresponding test case names are testPeopleInThatYear(), testPeopleInThatYearFail2().

Query Service:

1. For the QueryEvent method, we have one valid input and two invalid inputs
    a. For an input of a query without specifying the event name or with a non-existent event name, if the user can receive the corresponded message, the test is passed and vice versa. The corresponding test case is in queryEvent under QueryServiceTest class.
    b. For an input of a valid event, if the user can receive correct information about the event, the test is passed. The corresponding test case is in queryEvent under QueryServiceTest class.
2. For the QueryFacility method, we have one valid input and two invalid inputs
    a. For an input of a query without specifying the facility name or with a non-existent facility name, if the user can receive the corresponded message, the test is passed and vice versa. The corresponding test case is in queryFacility under QueryServiceTest class.
    b. For an input of a valid event, if the user can receive correct information about the facility, the test is passed. The corresponding test case is in queryFacility under QueryServiceTest class.

Balance Service:

1. For the addBalance method, we have two valid inputs and two invalid inputs
    a. For an input with a non-existent user, the response will show that the user doesn't exist. The corresponding test case is addBalanceInvalidUserTest.
    b. For an input with a negative amount of balance, the response will show that the amount is invalid. The corresponding test case is addBalanceInvalidAmountTest.
    c. For an input with a user who hasn't performed balance related performance, the service will create an entry in the table. The corresponding test case is addBalanceNewUserTest.
    d. For an input with a user who has an entry in the balance table, the service should perform the corresponding operation. The corresponding test case is addBalanceExistingUserTest.
2. For the subBalance method, we have two valid inputs and three invalid inputs
    a. For an input with a non-existent user, the response will show that the user doesn't exist. The corresponding test case is subBalanceInvalidUserTest.
    b. For an input with a negative amount of balance, the response will show that the amount is invalid. The corresponding test case is subBalanceInvalidAmountTest.

    c. For an input with an amount greater than the account has, the response will show that the amount is invalid. The corresponding test case is subBalanceInsufficientTest.
    d. For an input with a user who hasn't performed balance related performance, the service will create an entry in the table. The corresponding test case is subBalanceNewUserTest.
    e. For an input with a user who has an entry in the balance table, the service should perform the corresponding operation. The corresponding test case is subBalanceExistingUserTest.
3. For the addQuickPass method, we have two valid inputs and two invalid inputs
    a. For an input with a non-existent user, the response will show that the user doesn't exist. The corresponding test case is addQuickPassInvalidUserTest.
    b. For an input with a negative amount of QuickPass, the response will show that the amount is invalid. The corresponding test case is addQuickPassInvalidAmountTest.
    c. For an input with a user who hasn't performed QuickPass related performance, the service will create an entry in the table. The corresponding test case is addQuickPassNewUserTest.
    d. For an input with a user who has an entry in the QuickPass table, the service should perform the corresponding operation. The corresponding test case is addQuickPassExistingUserTest.
4. For the subQuickPass method, we have two valid inputs and three invalid inputs
    a. For an input with a non-existent user, the response will show that the user doesn't exist. The corresponding test case is subQuickPassInvalidUserTest.
    b. For an input with a negative amount of QuickPass, the response will show that the amount is invalid. The corresponding test case is subQuickPassInvalidAmountTest.
    c. For an input with an amount greater than the account has, the response will show that the amount is invalid. The corresponding test case is subQuickPassInsufficientTest.
    d. For an input with a user who hasn't performed QuickPass related performance, the service will create an entry in the table. The corresponding test case is subQuickPassNewUserTest.
    e. For an input with a user who has an entry in the QuickPass table, the service should perform the corresponding operation. The corresponding test case is subQuickPassExistingUserTest.
5. For the queryBalance method, we have two valid inputs and one invalid inputs
    a. For an input with a non-existent user, the response will show that the user doesn't exist. The corresponding test case is queryInvalidUserTest.

b. For an input with a user who hasn't performed balance related performance, the service will create an entry in the table. The corresponding test case is queryNewUserTest.
c. For an input with a user who has an entry in the balance table, the service should perform the corresponding operation. The corresponding test case is queryExistingUserTest.

Rating Service:

1. For the rateFacility method, we have one valid input and two invalid inputs.
    a. For an input of a query without specifying the facility name or with a non-existent facility name, if the user can receive the corresponded message, the test is passed and vice versa. The corresponding test case is in rateTest1 and rateTest2
    b. For an input of a valid facility name, the service should perform a rate change operation as described. The corresponding test case is in rateTest0.

QuickPass Service:

2. For the addQuickPass method, we have one valid input and three invalid inputs.
    a. For an input of a non-existent facility name, the service should respond with a corresponding error. The test case is in addQCTest1.
    b. For an input with an existent Quick Pass id, the user should respond with a corresponding error. The test case is in addQCTest0.
    c. For an input with a non-existent user id, the service should respond with a corresponding error. The test case is in addQCTest2.
    d. For an input with a valid Quick Pass, the service will register this appointment and respond with OK. The test case is in addQCTest3.
3. For the deleteQuickPass method, we have one valid input and three invalid inputs.
    a. For an input of a non-existent Quick Pass id, the service should respond with a corresponding error. The test case is in addQCTest1.
    b. For an input without an existent userModel, the user should respond with a corresponding error. The test case is in addQCTest2.
    c. For an input with a user not having the privilege to delete the appointment , the service should respond with a corresponding error. The test case is in addQCTest3.
    d. For an input with a user , the service should respond with a corresponding error. The test case is in addQCTest4.
    e. For an input with a valid Quick Pass id and proper user information, the service will register this appointment and respond with OK. The test case is in addQCTest0.

Appointment Service:

1. For the addAppointment method in the service part, there is 1 kind of valid input and 5 kinds of invalid inputs.

a. For a null appointment input, the service will throw a corresponding exception and this exception will be caught by test or in the real implementation will be caught and handled by the controller part. The test case is in addEmptyAppointmentTest.
b. For an appointment which already exists which is indicated by an already existing appointment id in addAppointment, the service will throw a corresponding exception and this exception will be caught by test or in the real implementation will be caught and handled by the controller part. The test case is in addDuplicateAppointmentTest.
c. For an appointment which tries to make an appointment to a nonexistent event, the service will throw a corresponding exception and this exception will be caught by test or in the real implementation will be caught and handled by the controller part. The test case is in addWrongEventAppointmentTest.
d. For an appointment associated with a nonexistent user, the service will throw a corresponding exception and this exception will be caught by test or in the real implementation will be caught and handled by the controller part. The test case is in addWrongUserAppointmentTest.
e. For an appointment that tries to book full events, the service will throw a corresponding exception and this exception will be caught by test or in the real implementation will be caught and handled by the controller part. The test case is in addBusyEventAppointmentTest.
f. For a valid appointment, the service will add this appointment and adjust remaining positions in the event in the database and return the appointment id. The test case is in addAppointmentTest.

2. For the updateAppointment method in the service part, there are 2 kinds of valid input and 5 kinds of invalid inputs.
    a. For a null appointment input, the service will return a corresponding error. The test case is in updateNullAppointmentTest.
    b. For updating an appointment not existed before, the service will return a corresponding error. The test case is in updateNotExistAppointmentTest.
    c. For updating an appointment associated with a nonexistent event, the service will return a corresponding error. The test case is in updateWrongEventAppointmentTest.
    d. For updating an appointment associated with a nonexistent user, the service will return a corresponding error. The test case is in updateWrongUserAppointmentTest.
    e. For updating an appointment to book a full event that is not booked by this appointment before, the service will return a corresponding error. The test case is in updateFullEventAppointmentTest.

   f. For valid updating an appointment to another event, the service changes the corresponding records associated with the appointment and event. Then the service gives out a successful message. The test case is in updateAppointmentTest.
   g. For valid updating an appointment to the same event, the service changes the corresponding records in the appointment and then gives out a successful message. The test case is in updateSameEventAppointmentTest.
3. For the deleteAppointment method in the service part, there are 2 kinds of valid input and 3 kinds of invalid inputs.
   a. For a null appointment id input, the service will return a corresponding error. The test case is in deleteNullIdAppointmentTest.
   b. For a delete request for a nonexistent appointment, the service will return a corresponding error. The test case is in deleteNotExistAppointmentTest.
   c. For a request from a visitor (not a manager role) having a different id from the appointment owner, the service will return a corresponding error. The test case is in deleteNotSameUserAppointmentTest.
   d. For valid input from the visitor, the service will delete the corresponding records in the database and return a successful message. The test case is in deleteAppointmentTest.
   e. For valid input from the manager, the service will delete the corresponding records in the database and return a successful message. The test case is in deleteManagerAppointmentTest.
4. For the getAppointmentsByUserId method in the service part, there is 1 kind of valid input and 1 kind of invalid input.
   a. For input corresponding to a nonexistent user, the service will throw a corresponding exception and this exception will be caught by test or in the real implementation will be caught and handled by the controller part. The test case is in appointmentsUserNotExistsRecordsTest.
   b. For valid input, the service will return a list of appointments. The test case is in appointmentsRecordsTest.
5. For the addAppointment method in the controller part, there are 4 kinds of invalid inputs and 1 kind of valid input.
   a. For a valid input, the controller will add associated records in the database and return appointmentId. The test case is in addAppointmentControllerTest.
   b. For an invalid input with isLogin equals null, the controller will return an error message. The test case is in addAppointmentIsNullLoginControllerTest.
   c. For an invalid input with isLogin equals false, the controller will return an error message. The test case is addAppointmentIsFalseLoginControllerTest.
   d. For an invalid input with null login user, the controller will return an error message. The test case is addAppointmentNullLoginUserControllerTest.

    e. For other invalid inputs, the controller will handle the exception thrown by the service and return an error message. The test case is in addAppointmentErrorImplControllerTest.
6. For the updateAppointment method in the controller part, there are 2 kinds of valid inputs and 4 kinds of invalid inputs.
    a. For a valid input, the controller will update associated records in the database and return OK. The test case is in updateAppointmentControllerTest.
    b. For a valid input from the manager, the controller will update associated records in the database and return OK. The test case is in updateAppointmentManagerControllerTest.
    c. For an invalid input with isLogin equals false, the controller will return an error message. The test case is updateAppointmentFalseLoginControllerTest.
    d. For an invalid input with isLogin equals null, the controller will return an error message. The test case is updateAppointmentIsNullLoginControllerTest.
    e. For an invalid input with null login user, the controller will return an error message. The test case is updateAppointmentNullUserControllerTest.
    f. For a invalid input from a normal visitor user trying to update another user's appointment, the controller will return an error message. The test case is updateAppointmentWrongUserControllerTest.
7. For the deleteAppointmentId method in the controller part, there is 1 kind of valid input and 3 kinds of invalid inputs.
    a. For a valid input, the controller will update associated records in the database and return OK. The test case is in deleteAppointmentIdTest.
    b. For an invalid input with isLogin equals false, the controller will return an error message. The test case is deleteFalseIsLoginAppointmentIdTest.
    c. For an invalid input with isLogin equals null, the controller will return an error message. The test case is deleteNullIsLoginAppointmentIdTest.
    d. For an invalid input with null login user, the controller will return an error message. The test case is deleteNullUserAppointmentIdTest.
8. For the getAppointments method in the controller part, there are 4 kinds of invalid inputs and 1 kind of valid input.
    a. For a valid input, the controller will give back the appointment information of the login user. The test case is in getAppointmentsControllerTest.
    b. For an invalid input with isLogin equals false, the controller will return an error message. The test case is getFalseLoginAppointmentsControllerTest.
    c. For an invalid input with isLogin equals null, the controller will return an error message. The test case is getNullLoginAppointmentsControllerTest.
    d. For an invalid input with null login user, the controller will return an error message. The test case is getNullUserAppointmentsControllerTest.

e. For other invalid inputs, the controller will handle the exception thrown by the service and return an error message. The test case is getErrorAppointmentsControllerTest.

Ticket Service

1. For the *addTicket* method in the service part, there are 7 kinds of invalid inputs and 1 kind of valid input.
    a. For a valid input, the service will add a ticket record in the database and also deal with other associated records. The test case is in addTicketTest.
    b. For a user without enough money to buy a ticket, the service will throw an exception and this exception will be catched and handled in the controller part. The test case is in addNoEnoughMoneyTicketTest.
    c. For an input trying to add an existing ticket, the service will throw an exception and this exception will be catched and handled in the controller part. The test case is in addDuplicateTicketTest.
    d. For an input with userId not associated with any existing user, the service will throw an exception and this exception will be catched and handled in the controller part. The test case is in addUserNotExistTicketTest.
    e. For an input with a passed date in the ticket, the service will throw an exception and this exception will be catched and handled in the controller part. The test case is in addPassedTicketTest.
    f. For an input with null ticket, the service will throw an exception and this exception will be catched and handled in the controller part. The test case is in emptyTicketTest.
2. For the updateTicket method in the service part, there are 3 kinds of invalid inputs and 1 kind of valid input.
    a. For a valid input, the service will update associated records. The test case is in updateTicketTest.
    b. For a null ticket input, the service will return an error message. The test case is in updateEmptyTicketTest.
    c. For a invalid input trying to update a non-existent ticket, the service will return an error message. The test case is in updateUnmatchedTicketTest.
    d. For an input with a non-existent userId, the service will return an error message. The test case is in updateUserNotExistTicketTest.
3. For the deleteTicket method in the service part, there are 2 kinds of valid inputs and 3 kinds of invalid inputs.
    a. For a valid input from a normal visitor, the service will update associated records. The test case is in deleteTicketTest.
    b. For a valid input from a manager, the service will update associated records. The test case is in deleteManagerTicketTest.

    c. For a null ticketId input, the service will return an error message. The test case is in deleteNullIdTicketTest.
    d. For an input trying to delete a non-existent ticket, the service will return an error message. The test case is in deleteUnmatchedTicketTest.
    e. For an input from a normal visitor trying to delete another visitor's ticket, the service will return an error message. The test case is in deleteWrongUserTicketTest.
4. For the getTicketsByUserId method in the service part, there is only 1 kind of valid input since the controller part already filtered out all kinds of invalid inputs. We will discuss more in the controller part.
    a. For a valid input, the service will return all ticket information of the user. The test case is in ticketsRecordsTest.
5. For the addTicket method in the controller part, there are 5 kinds of invalid inputs and 1 kind of valid input.
    a. For a valid input, the controller will add associated records in the database and return ticketId. The test case is in addTicketControllerTest.
    b. For an invalid input with wrong email address, the controller will catch and handle the exception. The test case is in addTicketFailToSendMailTest.
    c. For an invalid input with isLogin equals null, the controller will return an error message. The test case is in addTicketNullLoginControllerTest.
    d. For an invalid input with isLogin equals false, the controller will return an error message. The test case is addTicketFalseLoginControllerTest.
    e. For an invalid input with null login user, the controller will return an error message. The test case is addTicketNullUserControllerTest.
    f. For other invalid inputs, the controller will handle the exception thrown by the service and return an error message. The test case is in addErrorTicketControllerTest.
6. For the updateTicket method in the controller part, there are 4 kinds of invalid inputs and 1 kind of valid input.
    a. For a valid input, the controller will update associated records in the database and return OK. The test case is in updateTicket.
    b. For an invalid input from the normal visitor who is not able to update the ticket, the controller will return an error message. The test case is in updateVisitorTicket.
    c. For an invalid input with isLogin equals false, the controller will return an error message. The test case is updateFalseLoginTicket.
    d. For an invalid input with isLogin equals null, the controller will return an error message. The test case is updateNullLoginTicket.
    e. For an invalid input with null login user, the controller will return an error message. The test case is updateNullUserTicket.

7. For the deleteTicket method in the controller part, there is 1 kind of valid input and 3 kinds of invalid inputs.
    a. For a valid input, the controller will update associated records in the database and return OK. The test case is in deleteTicket.
    b. For an invalid input with isLogin equals false, the controller will return an error message. The test case is deleteFalseLoginTicket.
    c. For an invalid input with isLogin equals null, the controller will return an error message. The test case is deleteNullLoginTicket.
    d. For an invalid input with null login user, the controller will return an error message. The test case is deleteNullUserTicket.
8. For the getTickets method in the controller part, there is 1 kind of valid input and 3 kinds of invalid inputs.
    a. For a valid input, the controller will give back the ticket information of the login user. The test case is in getTickets.
    b. For an invalid input with isLogin equals false, the controller will return an error message. The test case is getFalseLoginTickets.
    c. For an invalid input with isLogin equals null, the controller will return an error message. The test case is getNullLoginTickets.
    d. For an invalid input with null login user, the controller will return an error message. The test case is getNullUserTickets.

Weather Service

1. For the queryWeather method in the service part, there is 1 kind of valid input and 1 kind of invalid input.
    a. For a valid input, the service will return the weather information of the date requested. The test case is in queryWeatherTest.
    b. For an invalid input with wrong date for which we do not have weather information access, the service will throw an exception and this exception will be catched and handled in the controller part. The test case is in queryWrongDateWeatherTest.
2. For the queryWeather method in the controller part, there is 1 kind of valid input and 4 kinds of invalid inputs.
    a. For a valid input, the controller will give back the weather information. The test case is in queryWeatherControllerTest.
    b. For an invalid input with isLogin equals false, the controller will return an error message. The test case is queryFalseLoginWeatherControllerTest.
    c. For an invalid input with isLogin equals null, the controller will return an error message. The test case is queryNullLoginWeatherControllerTest.
    d. For an invalid input with null login user, the controller will return an error message. The test case is queryNullUserWeatherControllerTest.

e. For other invalid inputs, the controller will handle the exception thrown by the service and return an error message. The test case is queryErrorWeatherControllerTest.

Mail Service

1. For the sendTicketMessage method, we have a valid input and one invalid input.
    a. If the message is sent by mail successfully, the inputs of email address and text must be valid. The test case name is sendTicketSuccessfullyTest().
    b. If the message is failed to be sent by mail, the inputs of email address and text are invalid. The test case name is failToSendTicketTest().
2. For sendAnnouncementMessage method, we have a valid input and one invalid input.
    a. If the message is sent by mail successfully, the inputs of email address and text must be valid. The test case name is sendAnnoucementSuccessfullyTest().
    b. If the message is failed to be sent by mail, the inputs of email address and text are invalid. The test case name is failToSendAnnoucementTest().

The test suit is under the src/test/java/com/java/rollercoaster directory.

Part III

The report of branch coverage is in the target/site/jacoco/index.html. And we omit the scan for the files under pojo and dao, which is generated by mybatis. We also omit the scan for the files under model, because there are only getters and setters. The report of checkStyle is in the target/site/checkstyle.html and the report of spotbugs is in the target/site/spotbugs.html. And we also omit the scan for those files which are used for configurations such as webMvcConfig.java, MybatisGenerator and so on.

Part IV

We use travis as our CI tools and the configuration file is .travis.yml. And our report can be obtained from the ReadMe.md after you click the button.