# CptS 583 Software Quality

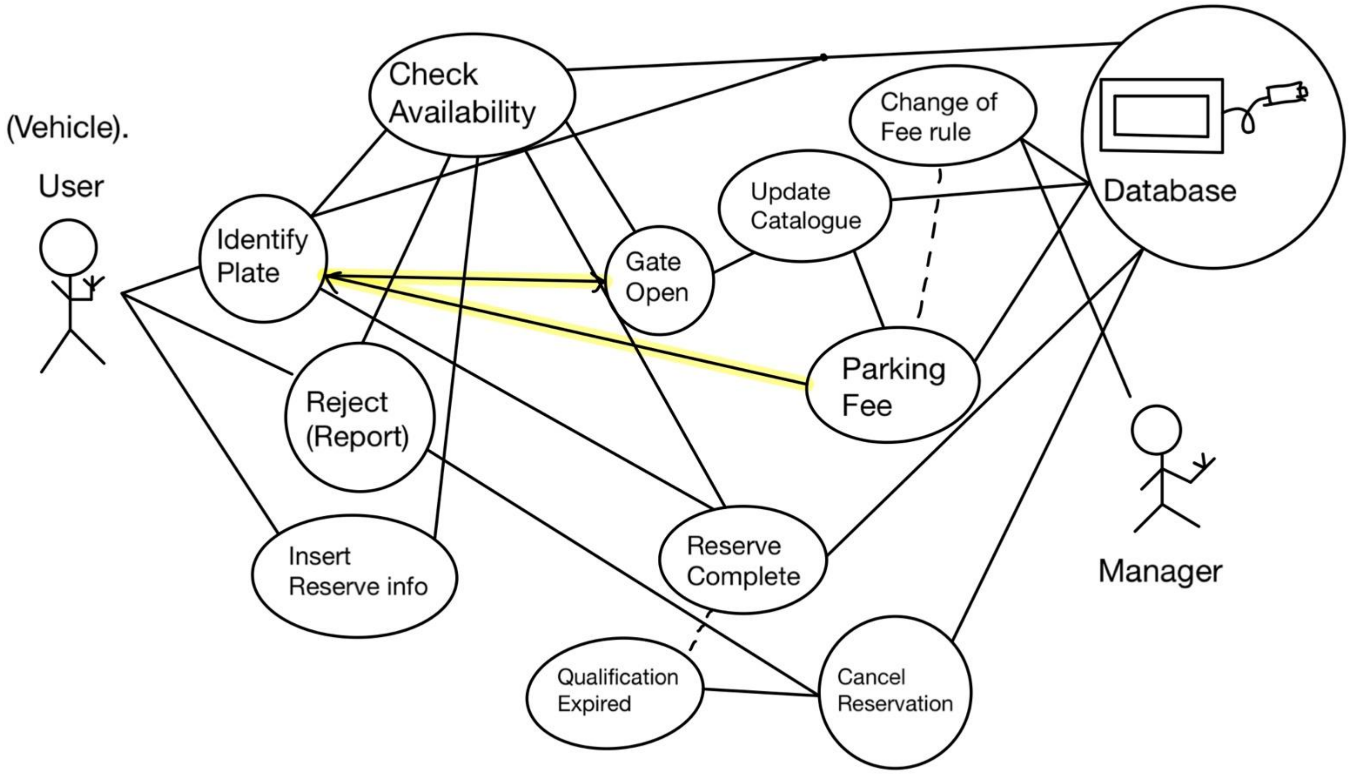
Team: SKS- SPS quality

Instructor: Haipeng Cai

Team members:

Hsueh-Jen Lih, Guang-Zheng Lee, Hsuan-Yu Chen, Ping-Wen Chen

## a. Use Case Diagram



## b. Use Case Elaboration

| | |
|---|---|
| **Use Case** | Idenfity Plate |
| **Actors** | User, Database |
| **Goal** | Identify car plate and import car information. |
| **Preconditions** | Stable detection system; Parking demand. |
| **Scenarios** | Potter wants to park his car and drive his car to the entery. |
| **Exceptions** | System malfunctioning; unreadable car plate. |

| | |
|---|---|
| **Use Case** | Check availability |
| **Actors** | Database |
| **Goal** | Check whether there are remaining parking spots. |
| **Preconditions** | Car plate confirmed; system working. |
| **Scenarios** | System searching for available parking spot. |
| **Exceptions** | System malfunctioning; Bug. |

| | |
|---|---|
| **Use Case** | Gate Open |
| **Actors** | Database(System) |
| **Goal** | Lift gate upon idenfitication of certain car. |
| **Preconditions** | Satisfy state "check availability". |
| **Scenarios** | Car drive through after gate lifted. |
| **Exceptions** | System malfunctioning. |

| | |
|---|---|
| **Use Case** | Update Catalogue |
| **Actors** | Database(System) |
| **Goal** | Update databse after gate lifted. |
| **Preconditions** | Request of "Gate Open" exist. |
| **Scenarios** | Update database. |
| **Exceptions** | System malfunctioning. |

| | |
|---|---|
| **Use Case** | Parking Fee(Pay) |
| **Actors** | User, Database |
| **Goal** | Calculate parking fee based on parking duration. User pays the calculated fee. |
| **Preconditions** | User is leaving. |
| **Scenarios** | User plans to leave the parking lot and trys to pay the fee. |
| **Exceptions** | Payment rule has been changed; System malfunctioning; Manager exception. |

| | |
|---|---|
| **Use Case** | Change of fee rule |
| **Actors** | Database, Manager |
| **Goal** | Modify calculation for parking fee. |
| **Preconditions** | Existing old method. |
| **Scenarios** | Manager meants to modify rule. |
| **Exceptions** | Unreasonable changing of fee rule |

| | |
|---|---|
| **Use Case** | Reject(Report) |
| **Actors** | User, Database |
| **Goal** | Notify user parking lot is out of parking spot. |
| **Preconditions** | Parking lot is full. |
| **Scenarios** | User receive the notification that the parking lot is full. |
| **Exceptions** | Parking spot exist. |

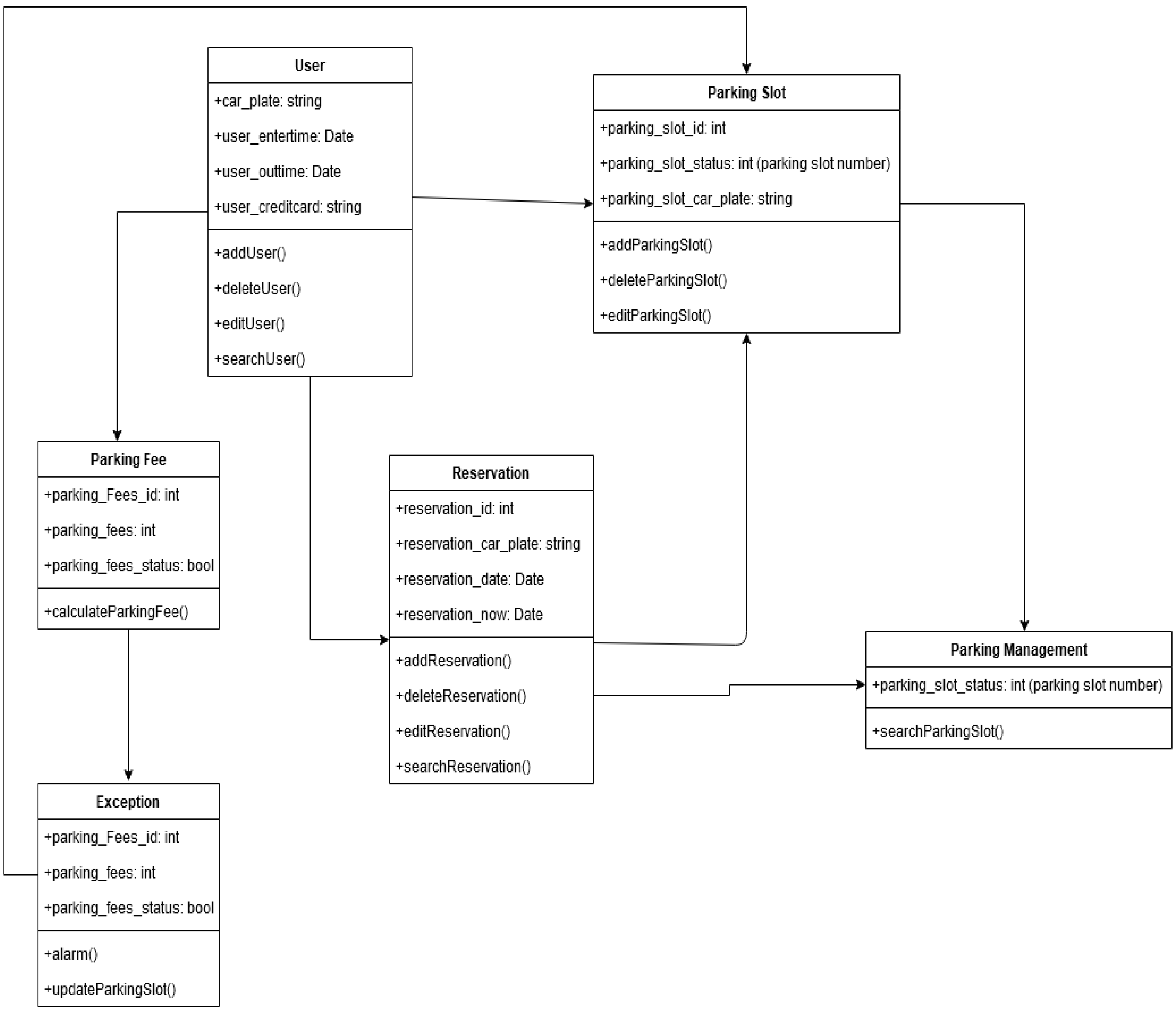
| | |
|---|---|
| **Use Case** | Insert reserve info |
| **Actors** | User, Database |
| **Goal** | Key in related information for reservation. |
| **Preconditions** | User plans to reserve parking spot for particular time period. |
| **Scenarios** | User is typing reserve information on the system. |
| **Exceptions** | Parking lot full; Duplicated reservation; The reservation is too close to the desire parking period. |

| | |
|---|---|
| **Use Case** | Reserve Complete |
| **Actors** | User, Database |
| **Goal** | Notify user that the reservation is complete. |
| **Preconditions** | Confirm parking spot available. |
| **Scenarios** | User complete the reservation and system create feedback. |
| **Exceptions** | Unstable internet connection |

| | |
|---|---|
| **Use Case** | Qualification Expired |
| **Actors** | Database |
| **Goal** | Avoid waste of parking spot. |
| **Preconditions** | Reservation expired; User does not appear. |
| **Scenarios** | System detected expired reservation. |
| **Exceptions** | Manager exception. |

| | |
|---|---|
| **Use Case** | Cancel Reservation |
| **Actors** | Database, User |
| **Goal** | Cancel Reservation and notify user. |
| **Preconditions** | Qualification expired. |
| **Scenarios** | System detected expired reservation. |
| **Exceptions** | Manager exception. |

## c. UML class diagram



## Quality Plan

### A. Quality goals and metric

| Product Quality | Quality Goals | Quality Metrics | Strategy |
|---|---|---|---|
| Availability | System working 24/7 | Error rate lower than 5% per month | Function testing |
| Reliability | System works correctly 24/7 | Accept 5% error rate per month | Workload stress testing |
| Robustness | System can detect types of car plate | No error exist | Testing different types of car plate |
| Learnability | This system can operate intuitively | A new user can understand this system by reading simple instruction | Remote usability testing |
| Usability | This system is extreme simple for users to use | This system interface has helpful simple instruction | Remote usability testing |
| Efficiency | This system generate result clear and quick | The system should return result less than 5s at most scenario(98%) | Unit test on general cases and special cases |
| Security | The system should not be compromised | Monthly audits should reveal fewer than 5 security issues KLOC | Unit test on general cases and special cases |
| Portability | This system can work for different kinds of parking lot. The installation time could be done in half of a day. | The system should be compatible to any other system configuration | SQL test |
| Process Quality | | | |
| Maintainability | The system could maintain easily | The source code should not contain lexical and design anti-patterns, be well documented, readable, and not contain any code smells. | Code review, Static code analysis tools |
| Testability | This system can be tested easily | Every class must have both white and black box testing, with integration/mutation testing where possible | Unit and mutation testing tools |

## Priority of quality goals(high->low):

1.Availability, Reliability
2.Learnability, Usability, Security
3.Efficiency, Maintainability
4.Testability, Robustness
5.Portability

Additional notes:

Quality goals have been discussed and ordered in reasonable way we want. We put **availability** and **reliability** at the first place since this is the most important thing in designing software. Nowadays, companies seek for designing user-friendly interface. Therefore, we move **learnability** and **usability** to second place. Different from most software, we decided to move **security** concern to a higher priority since our system only accept credit card and electronic payment. Beside these issues, **efficiency** would lower time consuming in any cases which brings higher income to the company. **Maintainability** applies the same rules above. The rest are the **testability**, **robustness**, and **portability**. These qualities do not directly impact users, and we expect no modification in the future.

## B. Costs of Quality

| **Task Name** | **Estimated Effort(hrs)** | **Implementation** | **Evaluation** | **Prevention** | **Rework** |
|---|---|---|---|---|---|
| Project Planning | 10 | 8 | 1 | 0 | 2 |
| Infrastructure setup | 5 | 3 | 1 | 0 | 2 |
| UI design | 7 | 5 | 3 | 0 | 2 |
| Initial Project development and Unit Testing | 35 | 25 | 5 | 3 | 5 |
| Quality Assurance-Integration/E2E testing | 30 | 28 | 5 | 2 | 7 |
| Acceptance Testing | 15 | 12 | 5 | 3 | 3 |
| Software Bug Correction | 10 | 8 | 5 | 0 | 2 |
| **Total** | 112 | 89 | 25 | 8 | 23 |