**Software Requirements Specification**

Version 0.13

for

BookMe

Prepared by

|  |  |  |  |
| --- | --- | --- | --- |
| Emir Bozer  Nikolas De Vigne Blanchet  Ahmad Hyjaz Loudin  Mary Psaroudis  Leo Yu | | 26424724  27189877  27179294  27209193  27036736 | emir.bozer@gmail.com  nikdvb@gmail.com  hyjaz.loudin@gmail.com  marypsaroudis@gmail.com  yuleo@outlook.com |
|  |  |  |

|  |  |
| --- | --- |
| Instructor: | Constantinos Constantinides |
| Course: | SOEN343: Software Architecture and Design I |
| Date: | 23/11/2016 |

**Document history**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 15/10/2016 | 0.01 | Added Intro, Description and DM | Leo Yu |
| 15/10/2016 | 0.02 | Added UC model, actor goal list | Nikolas De Vigne Blanchet |
| 16/10/2016 | 0.03 | Updated DM and added a box diagram for product perspective | Leo Yu |
| 18/10/2016 | 0.04 | Added External Interfaces | Emir Bozer |
| 18/10/2016 | 0.05 | Added Maintainability, Portability | Emir Bozer |
| 20/10/2016 | 0.06 | Updated the DM | Leo Yu |
| 21/10/2016 | 0.07 | Design constraints | Ahmad Hyjaz Loudin |
| 21/10/2016 | 0.08 | Fully-dressed Use Cases | Ahmad Hyjaz Loudin |
| 22/10/2016 | 0.09 | Added State diagrams description and diagrams | Maria Psroudis |
| 23/10/2016 | 0.10 | Added SSDs and missing contract | Emir Bozer |
| 12/11/2016 | 0.11 | Updated Reliability, Usability, and added Efficiency | Maria Psroudis |
| 12/11/2016 | 0.12 | Formatting | Leo Yu |
| 13/11/2016 | 0.13 | Formatting | Leo Yu |
|  |  |  |  |

**Table of contents**

[1. Introduction 5](#_Toc437438919)

[Purpose 5](#_Toc437438920)

[Scope 5](#_Toc437438921)

[Definitions, acronyms, and abbreviations 6](#_Toc437438922)

[References 6](#_Toc437438923)

[2. Overall description 7](#_Toc437438924)

[Product perspective 7](#_Toc437438925)

[Product functions 8](#_Toc437438926)

[User characteristics 9](#_Toc437438927)

[Constraints 9](#_Toc437438928)

[Assumptions and dependencies 9](#_Toc437438929)

[3. Specific requirements 10](#_Toc437438930)

[External interfaces 10](#_Toc437438931)

[Functionality 10](#_Toc437438932)

[Actor goal list 10](#_Toc437438933)

[Use case view 11](#_Toc437438934)

[Reliability 16](#_Toc437438935)

[Usability 16](#_Toc437438936)

[Efficiency 16](#_Toc437438937)

[Maintainability 16](#_Toc437438938)

[Portability 16](#_Toc437438939)

[Design constraints 16](#_Toc437438940)

[4. Analysis Models 17](#_Toc437438945)

Domain Model…………………………………………………………………………………………………………………………………..16

System Sequence Diagram………………………………………………………………………………………………………………..20

System Operations…………………………………………………………………………………………………………………………….23

Contract Operations………………………………………………………………………………………………………………………….23

State Diagrams………………………………………………………………………………………………………………………………….25

**List of figures**

Figure 1. Block Diagram of BookMe………………………………………………………………………………………......6

Figure 2. Use Case Model………………………………………………………………………………………………………....10

Figure 3. BookMe Domain Model……………………………………………………………………………………………..16

Figure 4. Room Reservation SSD……………………………………………………………………………………………….19

Figure 5. Add to Waiting List SSD……………………………………………………………………….………………………20

Figure 6. Modify Reservation SSD………………………………………………………………………………………….….20

Figure 7. Cancel Reservation SSD…………………………….………………………………………………………………..21

Figure 8. View Schedule SSD……………………………………………………………………………………………………..21

Figure 9. Room Reservation State Diagram……………………………………………………………………………….25

Figure 10. Modify Reservation State Diagram……………………………………………………………………………26

Figure 11. Cancel Reservation State Diagram…………………………………………………………………………….26

Figure 12. View Schedule State Diagram……………………………………………………………………………………27

# Introduction

## Purpose

The purpose of this document is to clearly specify the software requirement for the system designated as BookMe. BookMe is a web application of an on-line conference room reservation system that will be used by Concordia University students. The Software Requirement Specification (SRS) Document is divided into three sections: Overall Description, Specific Requirements and Analysis Models. The Overall Description section provides contexts to the requirements of the on-line conference room reservation system and how different factors can affect the product. The Specific Requirements section lists in detail all functional requirement through use cases and non-functional requirement through the ISO/IEC 25010 standard. The Analysis Models section lists all UML models that will help visualize the structure and the interaction of the system based on the requirements.  This document is intended both developers and end- users to use as it provides a clear understanding of all the requirements of the system. For the end-users, they can use the document to review if all the requirements are met. For the developers, they can use the document to start designing the software architecture of system that will be used for implementation.

## Scope

The SRS Document describes the software system of the conference room reservation system that will be developed. It provides the functional and non-functional requirement to describe what the BookMe system is expected to do. Additionally, it provides some scenarios that will showcase how the system is supposed to interact with a user based on the requirements. Hence, this documentation is heavily influenced by the requirements of the software. If there is a change of requirements by the stakeholders, the document must be modified in order to reflect on the new changes.

## Definitions, acronyms, and abbreviations

ISO International Organization for Standardization.

IEC International Electronic Commission.

SRS Software Requirements Specifications.

UML Unified Modeling Language.

PostgreSQL An object-relational database system.

Python An interpreted, object oriented, high-level programming language with dynamic semantics.

Flask “[It] is a microframework for Python based on Werkzeug, Jinja2 and good intentions.” [1]

UC Use case

SSD System Sequence Diagram

## References

Provide a list of all documents referenced in the SRS.

[1] Flask. (2016) [Online]. Available: <http://flask.pocoo.org/>

# Overall description

## Product perspective

BookMe is a web application that allows the reservation of conference rooms in college facilities for registered users. However, BookMe is not a self-contained product as it requires access to a PostgreSQL database in order to fetch information for the system to execute. Thus, BookMe is responsible for authenticating the user, managing reservations, displaying reservations and requesting information from the PostgreSQL database. The PostgreSQL database is responsible for storing and providing both user data and reservation information to BookMe. The overall process is shown on Figure 1.

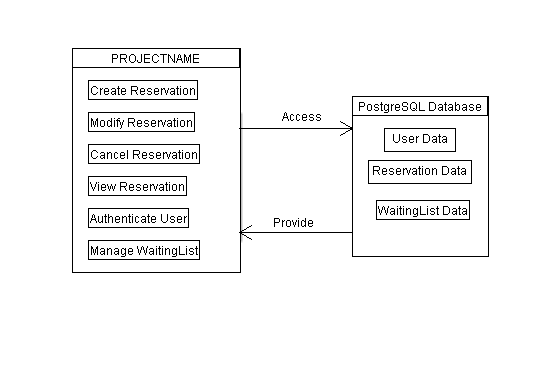


Figure . Block Diagram of BookMe

## Product functions

BooKMe contains five major system functions:

**Creating Reservation**

This major system function entails the reservation of a room through BooKMe. If the user is authenticated and the selected room is not currently occupied in the system, the user can reserve a room by providing the time and description of the reservation. BooKMe verifies if the user inputs are valid and if the room is available at the specified time. If it satisfy the conditions, the system creates the reservation and adds it to the registry.

**Viewing All Reservations**

This major system function describes the viewing all the reservations made in BookMe. An authenticated user can request the system to view all reservations. In return, BookMe displays all the reservations, which includes their timeslots and room number, back to the user.

**Modifying Reservation**

This major system function describes the modification of a reservation in BookMe. If the user is the reservation holder and the selected room is not currently occupied in the system, the user can modify his own reservations. BookMe verifies if the user inputs are valid and if the room is available at the new specified time. If it satisfy the conditions, the system modifies the reservation and updates it to the registry. Afterward, the waiting list will get updated.

**Cancelling Reservation**

This major system function describes the cancellation of a reservation in BookMe. If the user is the reservation holder and the selected room is not currently occupied in the system, the user can cancel his own reservations. In return, BookMe will delete the reservation and updates the registry. Afterward, the waiting list will get updated.

**Add to the Waiting List**

This major system function describes a user being added to a waiting list when a room is unavailable to be reserved at the specify time. The user can be added to a waiting list by providing the time and description of the reservation. BookMe verifies if the user inputs are valid. If the inputs, the system adds the reservation information to the waiting list.

## User characteristics

The intended users of BookMe are the students of Concordia University who have basic computer skills and knowledge to use the system.

## Constraints

* The inexperience with the programming language Python and the web framework Flask could dampen the development process of the project as time must be allocated to learn them.
* There is time constraint for this project. Thus, there is a possibility not all requirement can be met on time.
* The different programming environment could cause an issue later on as half the team is programming in a Mac environment and the other half is programming in a Windows environment.

## Assumptions and dependencies

* The BookMe web application shall be developed and executed on both Mac and Windows operating system.
* The BookMe web application shall be run on a local machine.
* Constant and reliable connection to the Internet is required in order to access the PostgreSQL database. Otherwise, a local database will be used.
* Users are assumed to be able to navigate and use the BookMe system perfectly.
* Assuming that all rooms in the system are the same.
* Assuming that the user would only need to modify the timeslot of their reservations.

# Specific requirements

## External interfaces

An external interface PostgreSQL is used to store all the reservations and users. The PostgreSQL stores the usernames, user passwords as well as all the reservations that has been made. When a User logs in to the BookMe application, it connects to the external interface to receive the login information of the user. When a reservation is made, it sends information about the reservation to the database. When a user view their reservations in the BookMe application, it connects to the external interface in order to receive, change and modify the reservation information.

* Hardware Interface

The Hardware Interface of the system is handled by Linux Centos 7 Operating System.

* Software Interface

The software is designed to run on Windows, Mac OSX or any Linux web server distribution. The software uses PostgreSQL to access its features. All the application features will be accessible through the internet website.

## Functionality

## Actor goal list

|  |  |
| --- | --- |
| Actor | Goal |
| User | * Login * Logout * Reserve a room * Cancel an existing reservation * Modify a reservation * View my reservations * View schedule * Add to waiting list |

## Use case view

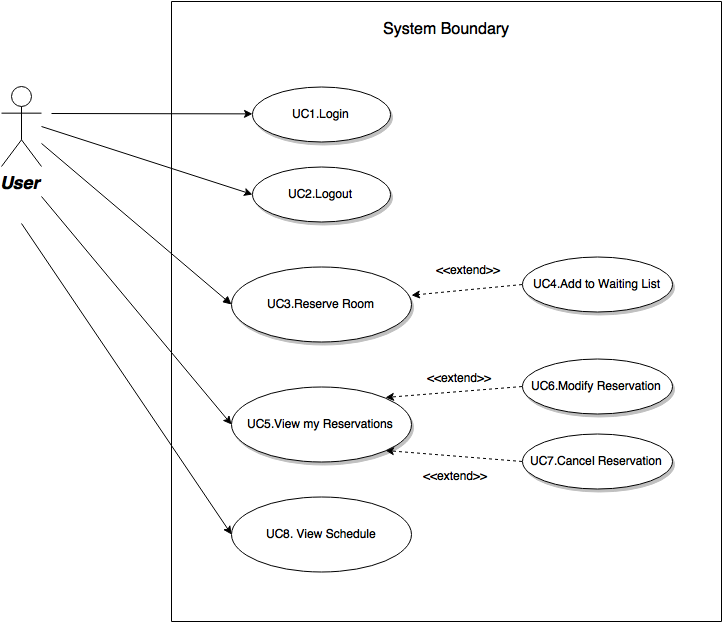


Figure . Use Case Model

|  |  |
| --- | --- |
| Use Case UC1: | Login |
| Primary Actor | User |
| Stakeholders and Interests: | User logs into the system |
| Preconditions: | * The user is currently not logged in |
| Postconditions: | * The user is logged in * The user has access to the functions of the system |
| Main success scenario: | 1. The user provides a valid username and password 2. The system indicates that the user has successfully logged in. |

|  |  |
| --- | --- |
| Use Case UC2: | Logout |
| Primary Actor: | User |
| Stakeholders and Interests | User logs out of the system |
| Preconditions | * The user is logged in |
| Postconditions | * The user has been logged out |
| Main success scenario | 1. The user requests to log out from the system 2. The system indicates that the user has successfully logged out. |

|  |  |
| --- | --- |
| Use Case UC3: | Reserve Room |
| Primary Actor: | User |
| Stakeholders and interests: | User reserves a room at specific date and time |
| Preconditions: | * User has been authenticated |
| Postconditions: | * The user has successfully reserved a room for a specific date and time |
| Main success scenario | 1. The user selects the desired room to be reserved 2. The system confirms the vacancy of the room selected and lock it 3. The user inputs a time, a date and a description to reserve the room 4. The system confirms the reservation of the room to the user 5. The user indicates that he/she is done with the reservation of the room process 6. The system responds with a confirmation and unlock the room 7. The user may repeat steps 1-6 until he/she is done with making reservation |

|  |  |
| --- | --- |
| Use Case UC4: | Add to waiting list |
| Primary Actor: | User |
| Stakeholders and interests: | User gets added to the waiting list |
| Preconditions: | * The user was unsuccessful reserving a room. |
| Postconditions: | * The user is added to the waiting list |
| Main success scenario: | 1. The user selects the desired room to be reserved 2. The system confirms the vacancy of the room selected and lock it 3. The user inputs a time, a date and a description to reserve the room 4. The system responds with the unavailability of the room at the specified time and date 5. The user requests to be added to the waiting list of that room with the same inputs when making the reservation 6. The system confirms the user is queued on the waiting list 7. The user indicates that he/she is done with being added to a waiting list process 8. The system responds with a confirmation and unlock the room 9. The user may repeat steps 1-8 until he/she is done |

|  |  |
| --- | --- |
| Use Case UC5: | View my reservations |
| Primary actor: | User |
| Stakeholders and Interests: | User views his/her reservations |
| Preconditions: | * User has been authenticated |
| Postconditions: | * The user is able to view all his/her reservations |
| Main success scenario | 1. The user requests to view his/her reservations 2. The system displays all of the user’s reservations |

|  |  |
| --- | --- |
| Use Case UC6: | Modify Reservation |
| Primary Actor: | User |
| Stakeholders and Interests: | User modifies an existing reservation |
| Preconditions: | * User has been authenticated * User is viewing his/her reservations |
| Postconditions: | * User was able to successfully modify date and time |
| Main Success Scenario: | 1. The user selects his/her reservation to be modified 2. The system confirms the vacancy of the selected room and lock it 3. The user modifies the time of the reservation 4. The system confirms the modification of the reservation to the user 5. The user indicates that he/she is done with the modification of the selected reservation 6. The system updates the waiting list and responds with a confirmation and unlock the room 7. The user may repeat steps 1-6 until he/she is done |

|  |  |
| --- | --- |
| Use Case UC7: | Cancel Reservation |
| Primary Actor: | User |
| Stakeholders and interests: | User cancels a specific reservation that he/she have made |
| Preconditions: | * A reservation has already been made. |
| Postconditions: | * The user has successfully cancelled the selected reservation |
| Main success scenario: | 1. The user selects one of his/her reservation to be cancelled 2. The user requests the cancellation of the selected reservation 3. The system confirms the cancellation of the reservation to the user 4. The user indicates that he/she is done with the cancellation process 5. The system responds with a confirmation and unoccupied the room |

|  |  |
| --- | --- |
| Use Case UC8: | View schedule |
| Primary actor: | User |
| Stakeholders and Interests: | User views all the reservations in the system |
| Preconditions: | * User has been authenticated |
| Postconditions: | * The user is able to view the available and occupied time slots and rooms |
| Main success scenario | 1. The user requests to view all reservations 2. The system displays all of the reservations |

## Reliability

The system will be available 20 hours a day, 7 days a week. User credentials will be stored and protected from potential hack threats and malicious software. Information input into the system will be saved except in the case of a power failure or unexpected crash.

## Usability

Any user with basic computer skills will be able to use the software application to perform the basic operations outlined in the use cases. The user interface will be clean and easy to use. Documentation will be provided for users who require assistance.

## Efficiency

The system should respond to user requests in no longer than 5 seconds. The system will be able to handle up to 50 users at a time.

## Maintainability

The software applications language (Python) is universal, therefore in terms of reusability there will be no issues for using the code in different systems. Since the software application does not control any physical hardware, there will be no issues in terms of modularity.  Also the software application can be easily modified without any defects because of the effectiveness and efficiency of the code.

## Portability

The software applications language (Python) is portable on Windows OS, Linux OS, Windows Server 2012 and Max OSX.

## Design constraints

In terms of programming languages, the project is required to be written in an object-oriented approach. Thus, flask is selected to be the main programming language of the project. For databases, queries to the relational database postgreSQL can be done through SQLAlchemy library provided by flask itself. However based on the business needs for this project it is required to manually implement object-relational structural and behavioral patterns. The system must be designed with restrictions such as: A room can be booked on a repeated time-slot, but only for some time period and a user may create only up to some maximum allowable number of reservations per week. Also, the design of the system must provide a safety, liveness and fairness.

# Analysis Models

**Domain Model**

The domain model is an abstract object model that showcases a static conceptual structure of the problem domain. Also known as the conceptual class diagram, the model illustrates concepts, associations between the concepts and attributes of concepts that are based on the functional requirement recorded in the use cases. Despite describing the overall project, the domain model doesn’t describe the software design of the project. In order words, the diagram doesn’t illustrate any system operations (methods or functions). In spite of that fact, all further analysis model can be traced back to the domain model, to which it can be traced back to the requirements. In the BookMe system, the domain model is composed of several conceptual objects: User, Registry, ReservationBook, Waiting, Reservation, Directory, Room and Timeslot.

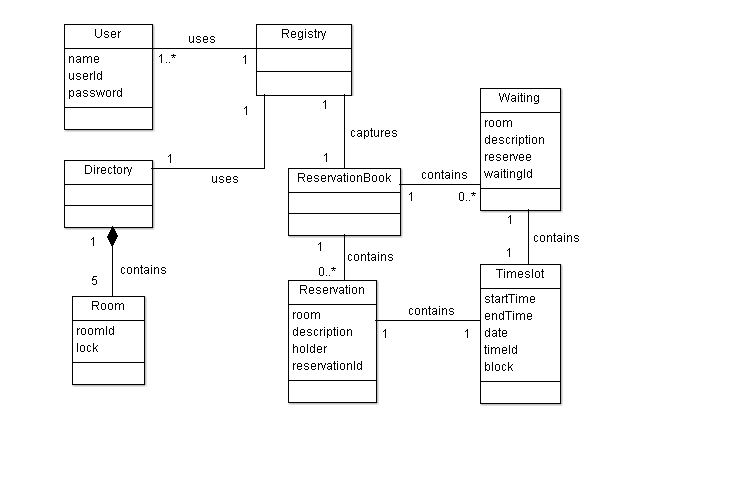


Figure . BookMe Domain Model

**User**

The User is an actor of the system that represents a Concordia student. The User can use the Registry to start a reservation session. Also, the User can view their Profile to see their reservations and the waiting list. Each User is characterized by a name, an id number, an academic program, an email and a password.

**Registry**

The Registry is an object that represents the controller of the system where it receives request from the User and assigns the appropriate responsibility to the system. The Registry captures the ReservationBook to start a reservation session and utilizes the Directory to get relevant information about the rooms. In addition, the Registry will update the Profile of the user with new information.

**ReservationBook**

The ReservationBook is an object that represent a reservation book that keeps a list of reservations in the system. It contains multiple instances of both Reservation and the Waiting objects. The ReservationBook object is used by the Registry to handle operation such as the creation, the modification, the deletion and the storing of both the Reservation and Waiting objects.

**Reservation**

The Reservation is an object that represents the reservation of a room at a specific timeslot. It is contained in the ReservationBook object. The Reservation object is characterized by a Room object, a description, a Timeslot object and the reservation holder’s name.

**Waiting**

The Waiting is an object that represents a user being put on a waiting list when a room at a specific timeslot is unavailable for reservation. It is contained in the ReservationBook object. The Waiting is characterized by a Room object, a Timeslot object, a description and the reservation holder’s name.

**Directory**

The Directory is an object that represents the directory of rooms for the reservation system. It contains multiple instances of a Room object. Thus, it is used by the Registry object to get information about the Rooms object.

**Room**

The Room is an object that represents the room that are going to be reserved in the system. It is contained in the Directory object. The Room object is characterized by room id number and its availability.

**Timeslot**

The Timeslot is an object that represents the time that the room is reserved for. It is contained in the Room object. It is characterized by the start time, the end time the date of the reservation and the block which represents the amount of one hour interval during the reservation.

**System Sequence Diagram**

System Sequence Diagram is a subtype of the sequence diagram that visually describes a use case, events of a use case, in the order of the execution. Use cases describe the user's interaction with the system. System Sequence Diagram also shows how the tasks that are related to uses cases are completed. They are represented with the notation SSD. In an SSD, there are external actors, messages, system responses and any loops that are needed.

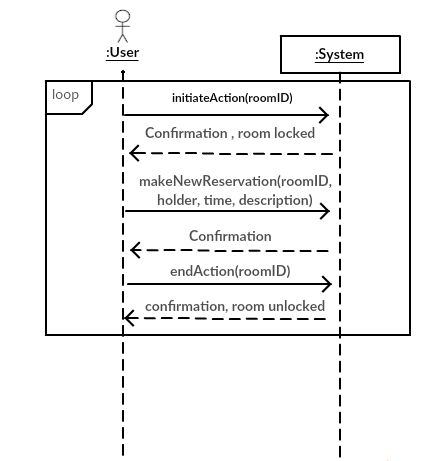


Figure . Reserve Room SSD

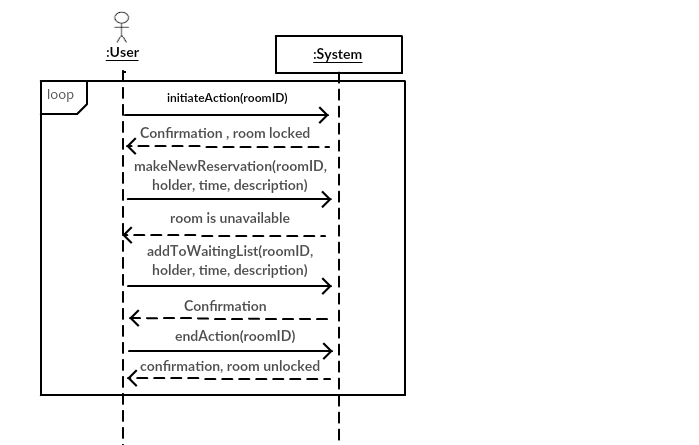


Figure . Add to Waiting List SSD

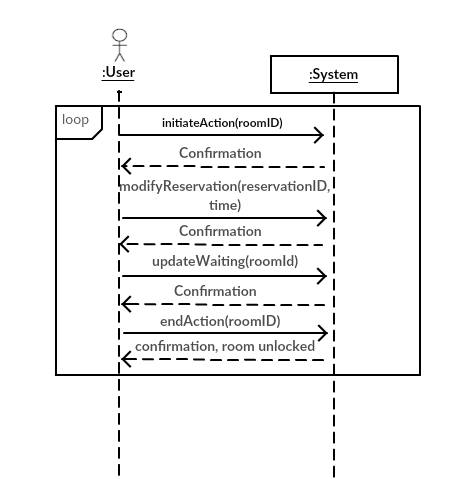


Figure . Modify Reservation SSD

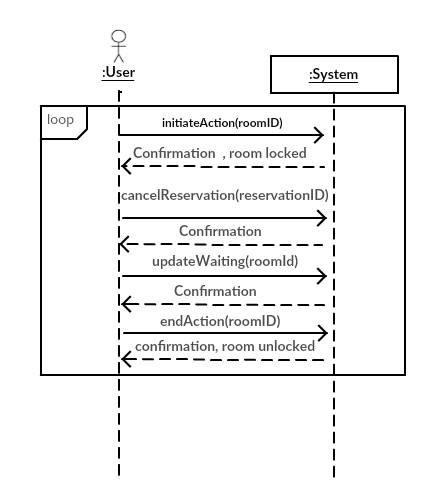


Figure . Cancel Reservation SSD

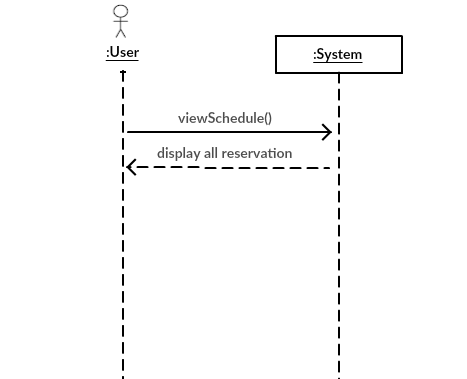


Figure . View Scedule SSD

**System Operations**

|  |
| --- |
| System Operations |
| initiateAction(roomId)  makeNewReservation(roomId, holder, time, description)  addToWaitingList(roomId, holder, time, description)  endAction(roomId)  modifyReservation(reservationId, time)  cancelReservation(reservationId)  viewSchedule()  updataWaiting(roomId) |

**Contract Operations**

|  |
| --- |
| **Contract CO1**: Make New Reservation  Operation: makeNewReservation(roomId, holder, time, description)  Cross References: Use Case Reserve Room  Preconditons:   * A ReservationBook session is underway   Postconditions:   * Reservation instance r was created * Attributes of r were initialized * r is added to ReservationBook |

|  |
| --- |
| **Contract CO2**: View Schedule  Operation: viewSchedule()  Cross References: Use Case View Schedule  Preconditons:   * User is logged in. * A ReservationBook session is underway.   Postconditions:   * A schedule is created from information contained in the ReservationBook |

|  |
| --- |
| **Contract CO3**: Modify Reservation  Operation: modifyReservation(reservationId, time)  Cross References: Use Case Modify Reservation  Preconditons:   * User is viewing his/her reservations * The user’s myReservations is non-empty   Postconditions:   * Reservation entry attributes updated |

|  |
| --- |
| **Contract CO4**: Cancel Reservation  Operation: cancelReservation(reservationId)  Cross References: Use Case Modify Reservation  Preconditons:   * User is viewing his/her reservations * The user’s myReservations is non-empty   Postconditions:   * Reservation entry removed from ReservationBook |

|  |
| --- |
| **Contract CO5**: Add to Waiting List  Operation:  addToWaitingList(roomId, holder, time,description)  Cross References: Use Case Add to Waiting List  Preconditons:   * The user has initiated a Reservation Session * The user has tried to make a new reservation, but the room is unavailable   Postconditions:   * A WaitingList instance w was created * Attributes of w were initialized * Instance w was added to user’s myWaitingList |

|  |
| --- |
| **Contract CO6**: Initiate Action  Operation:  initiateAction(roomId)  Cross References: Use Case Reserve Room, Add to Waiting List, Modify Reservation,  Cancel Reservation  Preconditons:   * The user has initiated a Reservation Session   Postconditions:   * The room’s lock is set to true |

|  |
| --- |
| **Contract CO7**: End Action  Operation:  endAction(roomId)  Cross References: Use Case Reserve Room, Add to Waiting List, Modify Reservation,                 Cancel Reservation  Preconditons:   * The initiateAction(roomId) operation has been called   Postconditions:   * The room’s lock is set to false |

|  |
| --- |
| **Contract CO8**: Update Waiting  Operation: updateWaiting(roomId)  Cross References: Modify Reservation, Cancel Reservation  Preconditons:   * The modifyReservation(reservationId) or cancelReservation(reservationId) operation has been called   Postconditions:   * If room is available at Waiting w’s timeslot, an instance of Reservation r is created * Reservation r’s attributes are initialized with the w’s attributes * Reservation r is added to reservationBook and w is removed from reservationBook |

**State Diagrams**

State diagrams are used to depict the behavior of objects in response to a series of external events. Each event triggers an object to move from one state to another. A state is the representation of a condition of an object moving through its behavior pattern. At any given moment, an object can only be in one state at a time. State diagrams are often used to help understand the life cycle of an object based on its preceding states and are used to model the dynamic flow of an object within a system. State diagrams can be applied to UML components such as classes and use cases. Moreover, state diagrams are effective in specifying the legal sequence of events. The following are state diagrams for four of the critical use cases which include: Room Reservation, Modify Reservation, Cancel Reservation, and View Reservations.

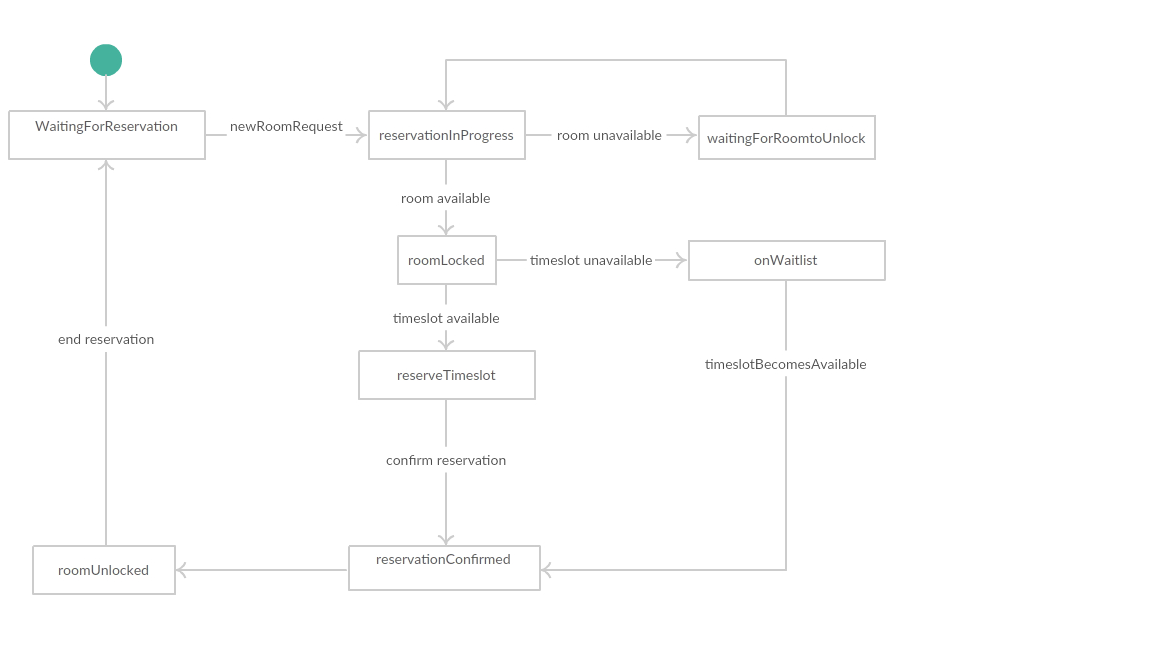


Figure . Room Reservation State Diagram

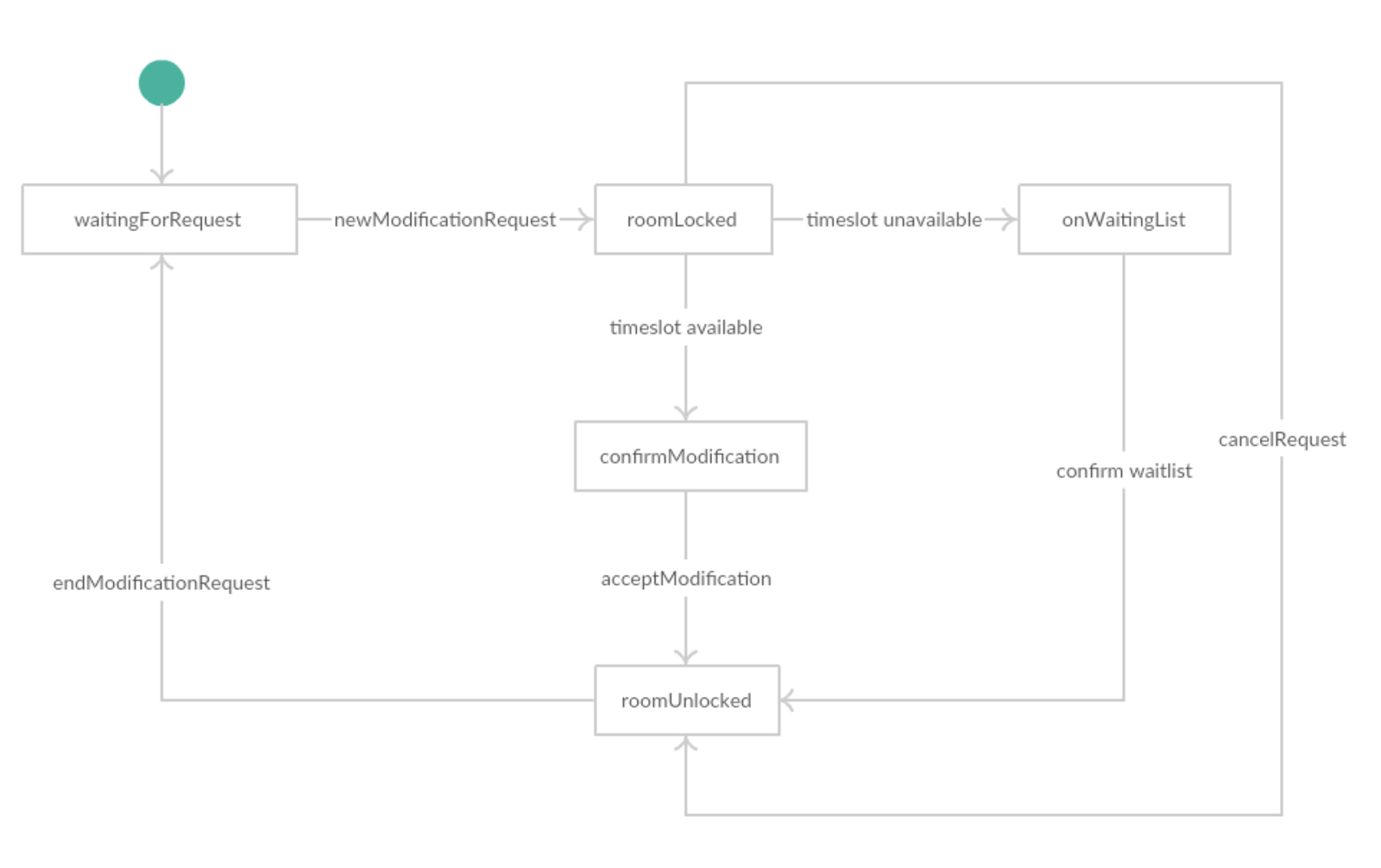
**

Figure . Modify Reservation State Diagram

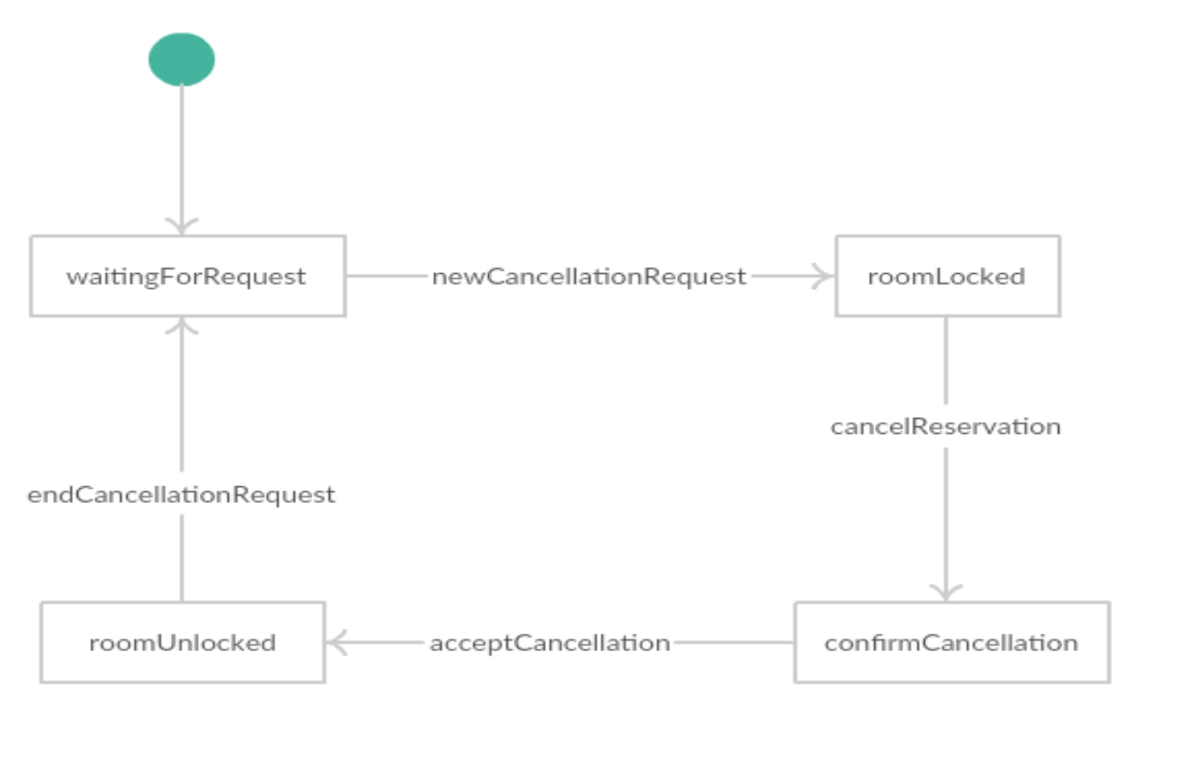
**

Figure . Cancel Reservation State Diagram

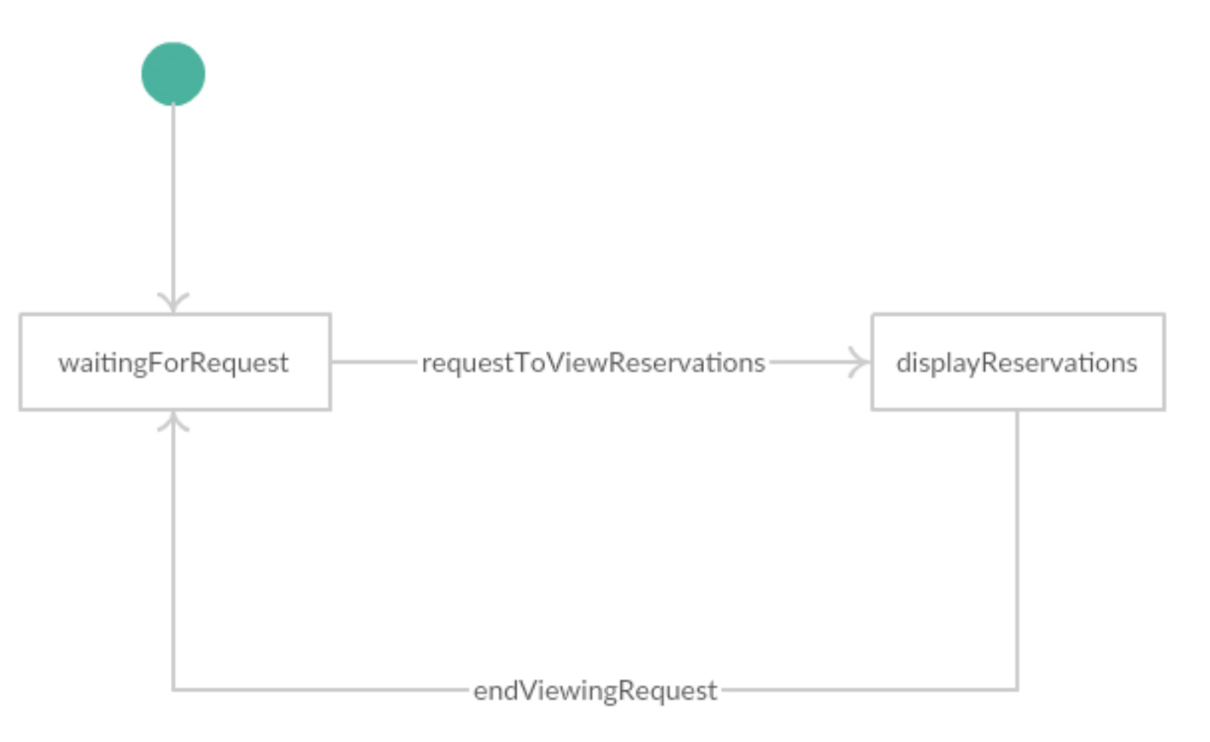
**

Figure . View Schedule State Diagram