**Counting Sheep**

Amir Sadiev, Imran Ahmed, Maclean Frazier , Walt Brady

Table of Contents

**1. Project Definition (**100 - 200 words**)** –*Group responsibility*

The purpose of this project is to make a hotel management system that allows both front and back end users to access and use the system. Since this is an application that will allow users to book hotel rooms, it will need friendly and easy to navigate GUI so that booking a room, requesting services and other accommodations is not an issue. The application will also support back end needs to employees whom follow up on their customer’s requests in order to make sure that all necessities are in place and issues are taken care of, as a key feature of this application allows users to directly make requests to the employees of the hotel they are staying in. All of this will be achieved by having a good application that is easy to use on both front and back ends, will allow easy modification, an up to date database that will make sure that there are no miscommunication anywhere in the application.

**2. Project Requirements** – *Group responsibility*

* Functional

The application will serve as a way for clients to book different hotels and rooms to try to get the best prices, deals, request services. It will serve as the gap between the hotel and the user and be the middleman where the customer will try to get the best deal. On the back send, it will allow operators of different hotels to manage their task all in one place. Customer could request service, cleaning and other responsibilities and the staff would get on it to get them done while they can also see other aspects of the business on the back side.

* Usability
  + User interface

There will be a GUI application that will allow both users and hotel employees to retrieve hotel information and make requests in an intuitive manner.

* + Performance

The program will load itself within a few seconds. The system will also load information from the database to the user quickly to prevent wasting the user’s time.

* System
  + Hardware

The system will be a desktop application and can be converted to a mobile application so users can use it anytime they want.

* + Software

This will be a desktop application connected to a mySQL database. It will use Java FX to display information to the user

* + Database

This system will use a mySQL database that updates in real time in response to user inputs.

* Security

Each user will have their own login credentials so that no one else can log in their account. Once the user logs in, there will be a security question the user will need to answer before they can access their entire account. They can reset their password using the question or another form of sensitive information like their social security number.

**3. Project Specification** – *Group responsibility*

* Focus / Domain / Area: Hotel Booking/Management for both front and back end users - Both the users or guests and the users as in management for the hotels will be able to use the application all in one place. You will not have to book in one place, and use another application to contact hotel staff or anything and the hotel will be able to contact you directly from the same application making it as smooth it can be.
* Libraries / Frameworks / Development Environment: This application will be programmed in JavaFX using the NetBeans IDE. It will use the Java Authentication and Authorization Service (JAAS) framework to maintain security in the whole system. The application will also use the JDBC API in order to connect the application to a database. The database itself will be created using mySQL in mySQL Workbench.
* Platform (Mobile, Desktop, Gaming, Etc): Desktop. The application will be a desktop application, and all the users will only be able to access it from a PC or laptop only. It can be developed into a mobile application but for now we will keep it PC only.
* Genre (Game, Application, etc): Travel/Management Application - The application is a travel application where it allows people to book rooms in hotels from anywhere. It will allow users to try to get the best rates possible and will allow the hotel owners to keep up with all the booking and accommodation from the website also.

**4. System – Design Perspective** – *Group responsibility*

* Identify subsystems – design point of view
  + - GUI

JavaFx - The reason I am using JavaFX is because it is a good language that has many features when designing a program. Since the other subsystems are being designed in java, it would be easier to connect all the subsystems if everything was written in the same language rather than different languages. JavaFx provides a lot of features on the design side that would help make a really good, easy to use GUI. It would be easy to use in the user side since the overall flow of the program would be smooth.

* + - Database

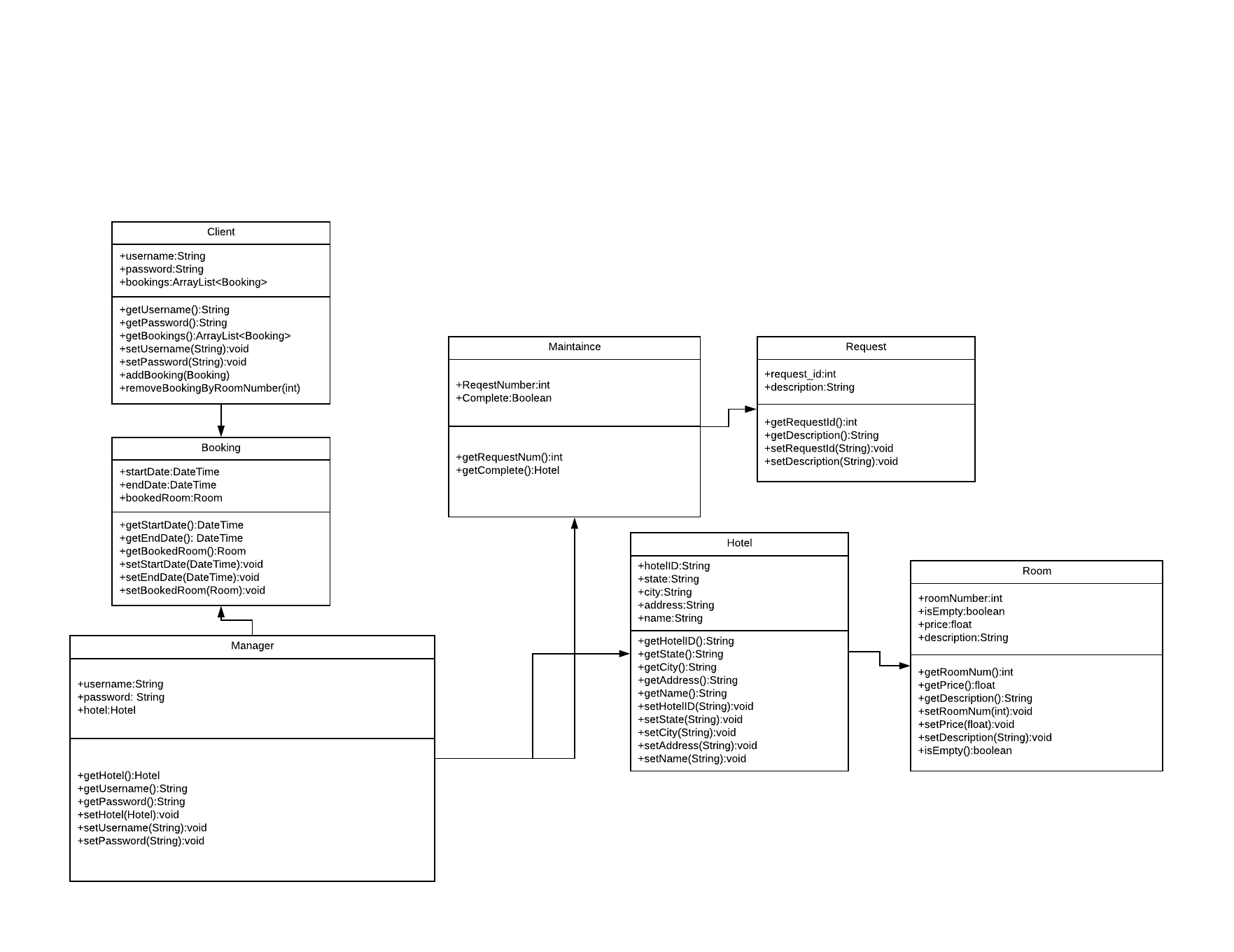
The database subsection is focused around developing a mySQL database and managing interactions between the main program and the database. The reason why mySQL is chosen over other options like noSQL is because mySQL is a relational database, meaning that querying data across multiple tables is easy and efficient and will prevent the database from slowing down the main application.

* + - Customer

Client side will use the UI to send requests such as choose date of bookings, list available hotels, as well as book rooms. These processes will send requests to the server for the respective information. Once processes are fully finished, the client will see a notification (confirmation) of completion. The whole project is based around Java language, Java FX (GUI) and mySQL database.

* + - Manager

The manager side, much like the customer subsystem, will be managed using Java. The subsystem will be designed to offer different services from the customer side, and will be linked to the database via room management. We are using java since it allows the creation of objects such as rooms and hotels that the manager will be able to easily manipulate through drop-down menus and text entry.

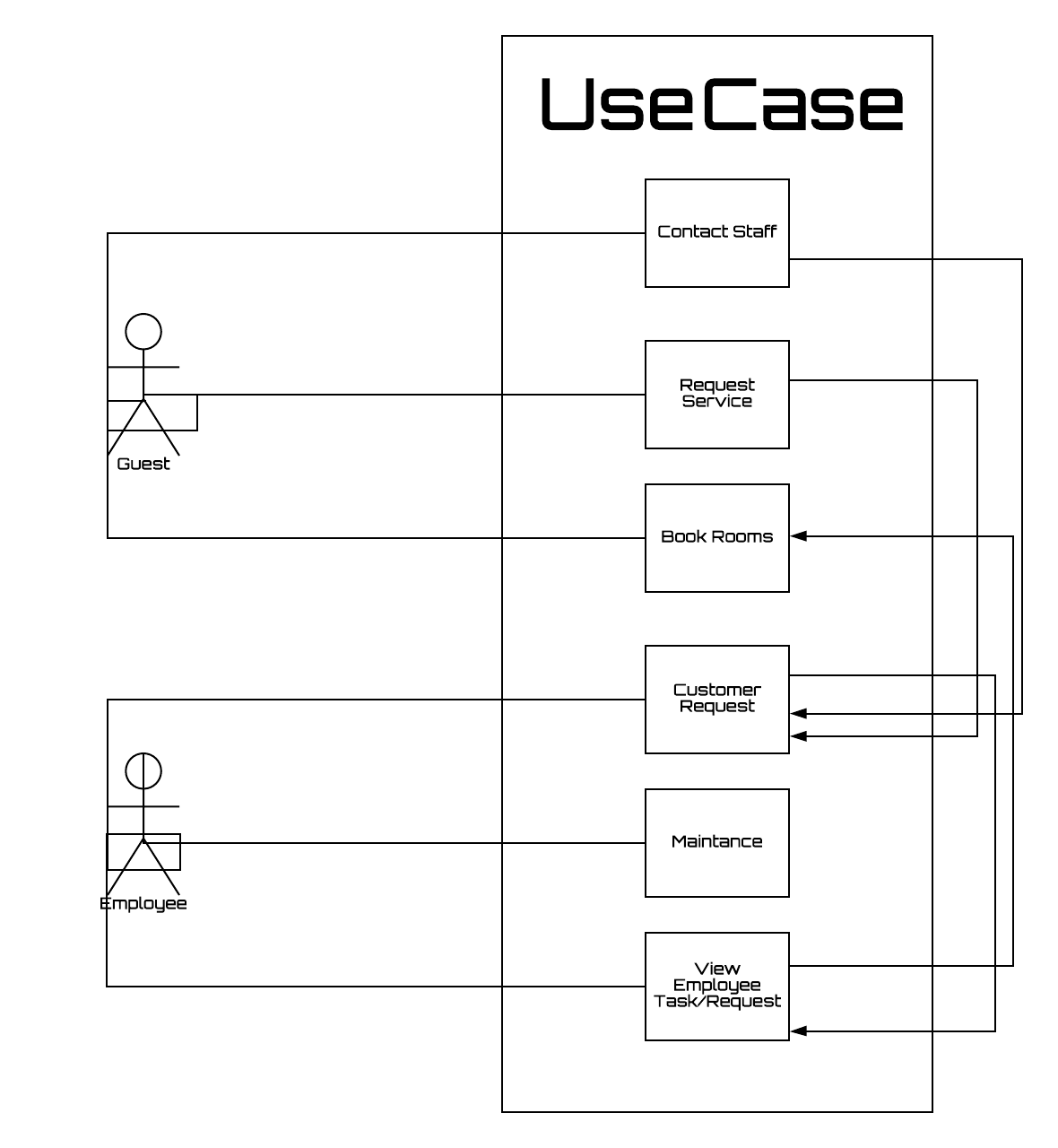


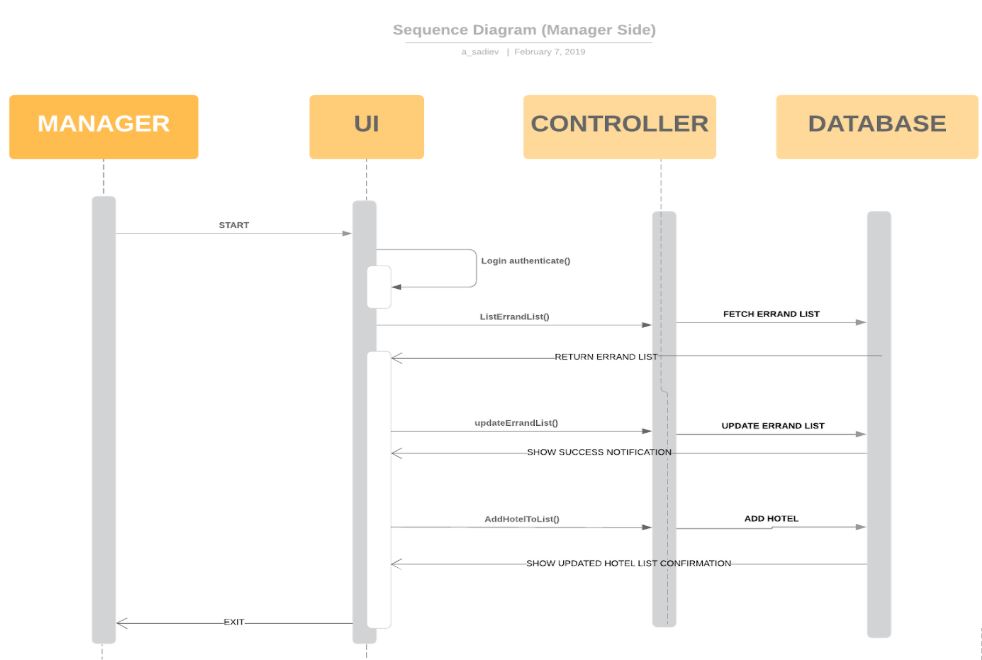
UML

Example UML for the Manager subsystem.

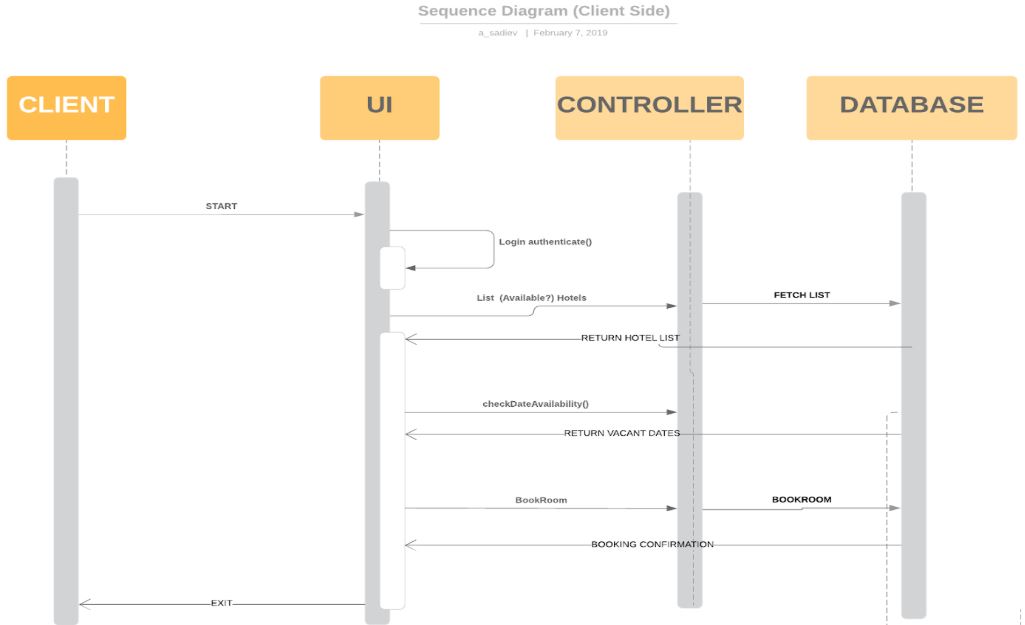
Use Case Diagram

The use case diagram represents both the guest and the employee. The Guest can contact the staff, request service and book rooms and then the employee would be notified and would take care of the request. One the employee side, they can follow up with customer request and/or complete their personal tasks assigned to them.





Sequence diagram for the manager side of the application, shows the sequence of events for a manager logging into the system.

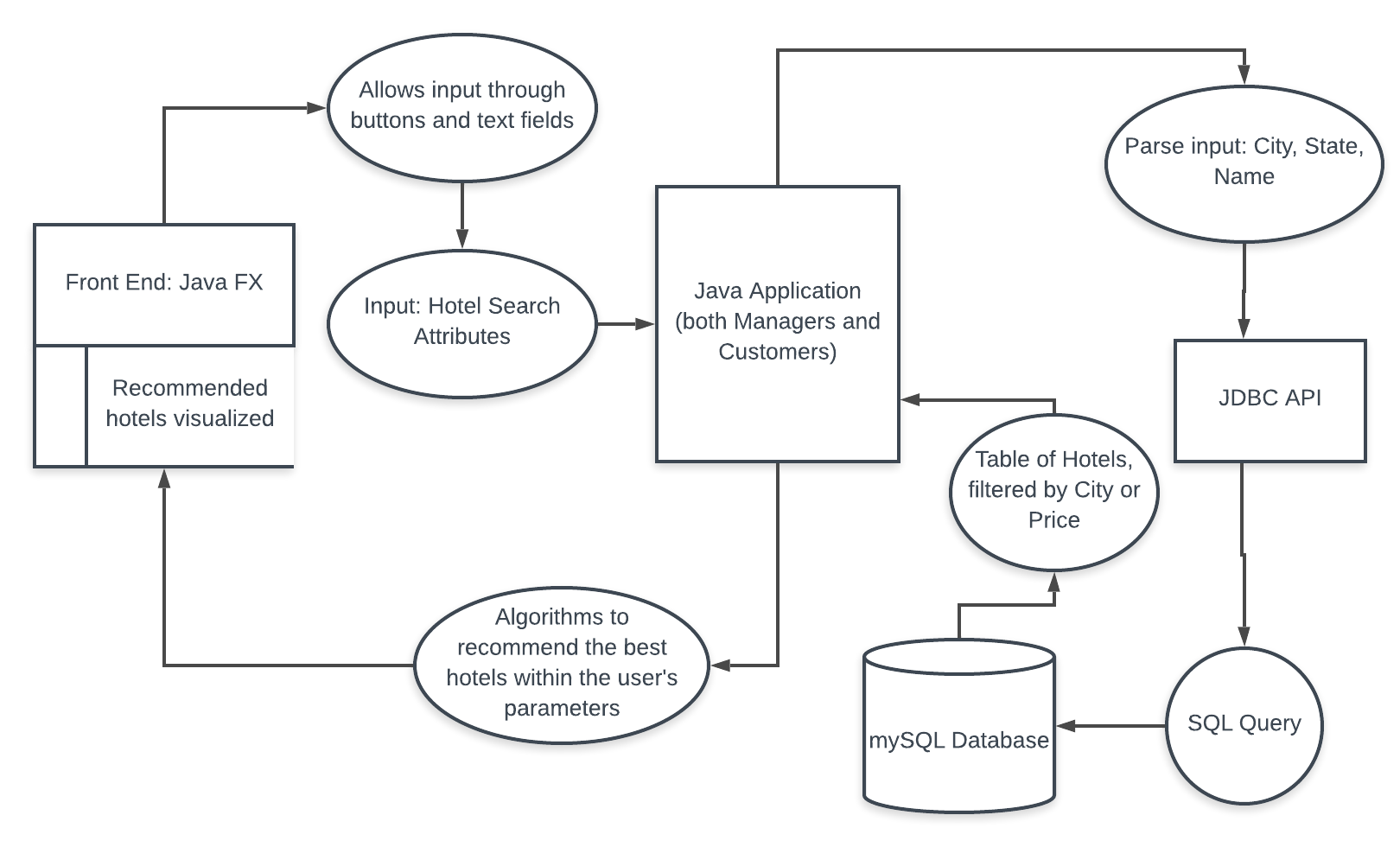


Sequence diagram for the customer side of the application, shows the sequence of events for a customer logging into the system.

* + Design choices

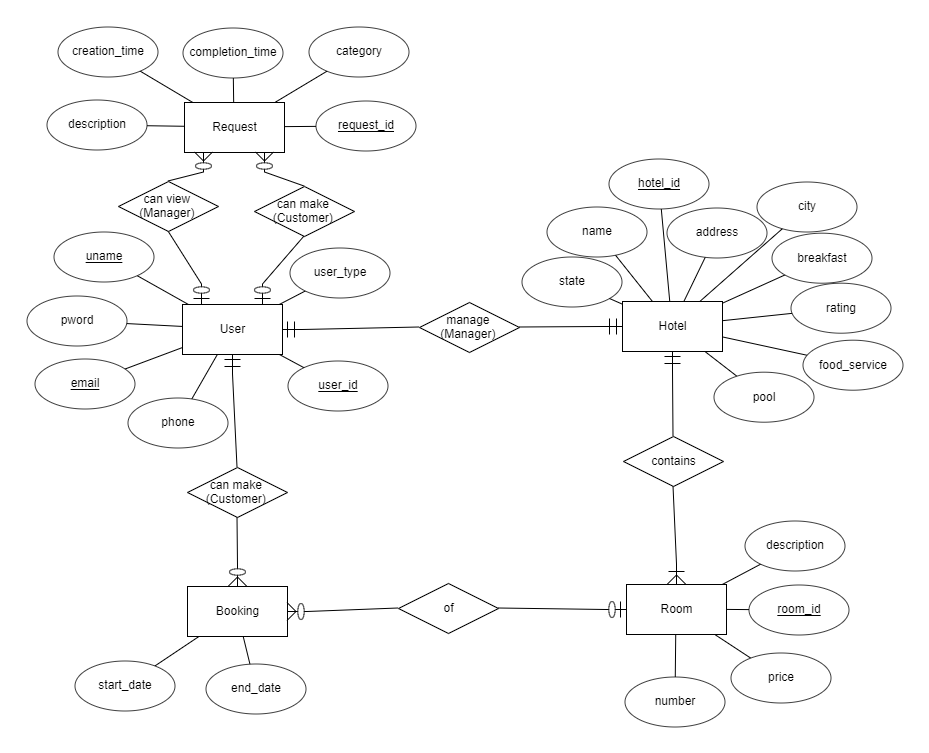
We are doing a desktop application for the moment. The desktop application will update in real time via the database, will make sure that the application is smooth running. The application can be incorporated into a mobile application down the road so that both users and employee can follow up on the go.

* Sub-Systems Communication (Diagram and Description)
  + Controls
  + I/O
  + DataFlow

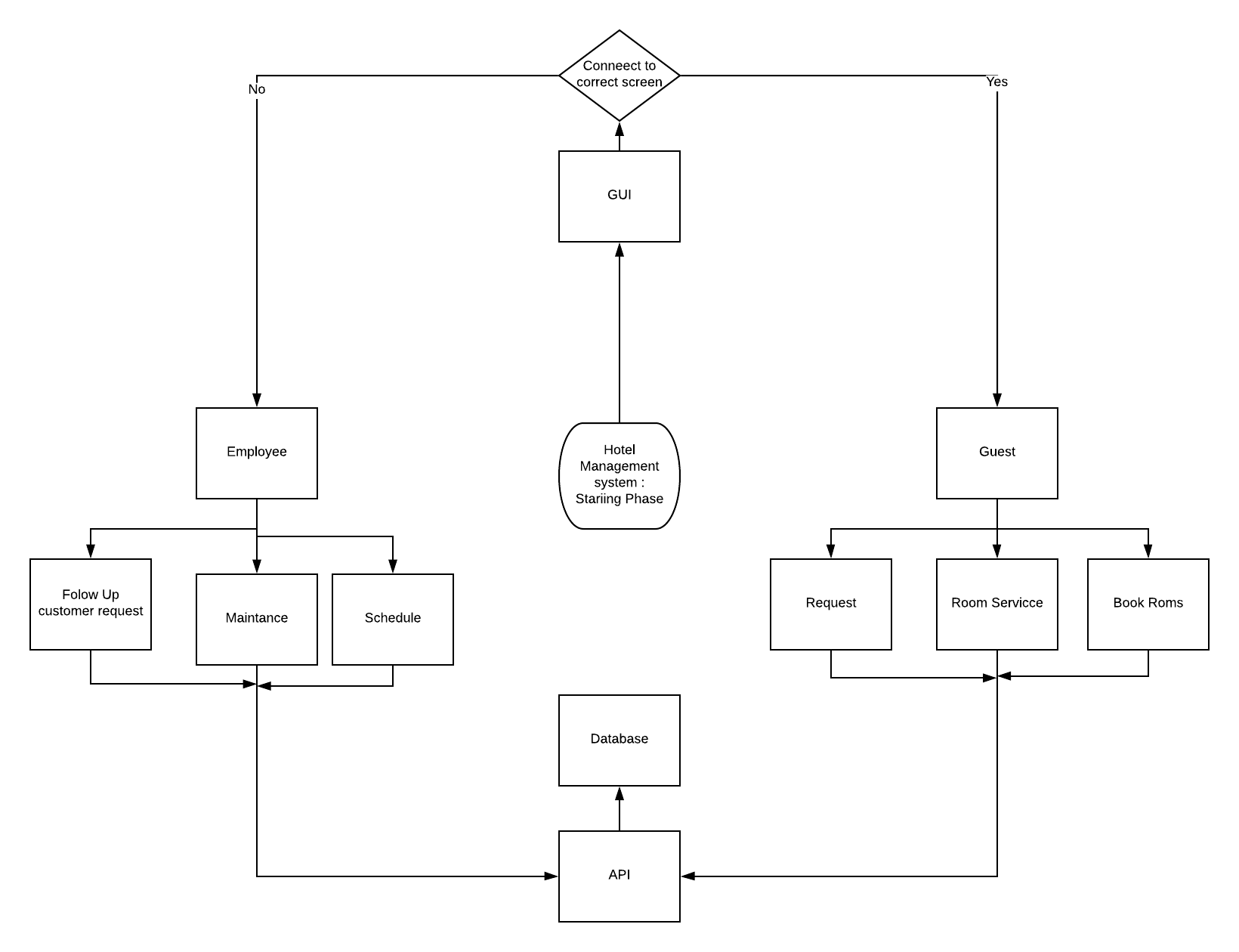


This diagram shows an example of how the data flows throughout the system, where a client searches for hotels with specific attributes.

* Entity Relationship Model (E-R Model)



* Overall operation - System Model
  + Simplified Sub-system to System interaction



**5. System – Analysis Perspective** – *Group responsibility*

* Identify subsystems – analysis point of view
  + GUI -

The GUI will be based of the login credentials when first logging in. Once logged in, the menu will display the appropriate menus according to who logged in and what all the options they have. Not all employees will have the same options since there will be different levels of employees and the same will go with the guest, they will have different menus according to their hotel plan.

* + Database

The database must be designed in such a way that it does not slow down the rest of the program. To do this in mySQL, indexes will be used. The database itself will be stored on Amazon’s RDS platform to allow quick access from multiple users at the same time. This allows both customers and managers to access valuable data as quickly as possible.

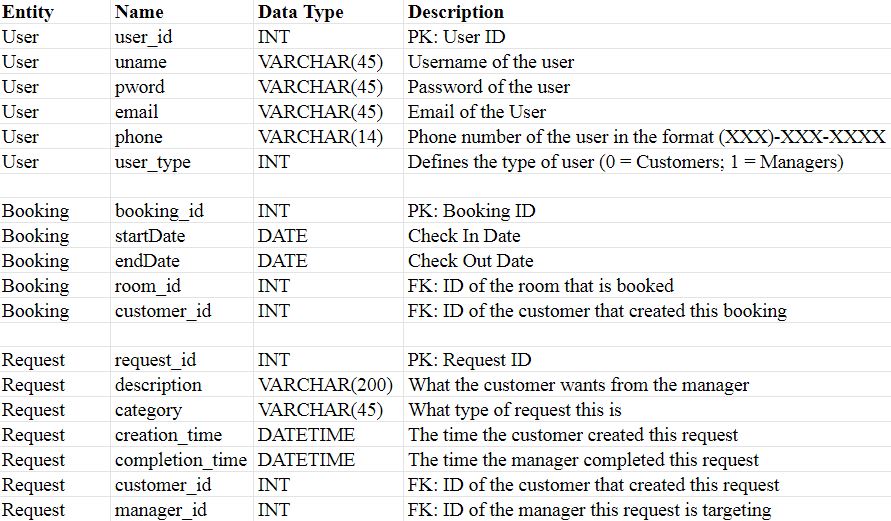
* + Customer

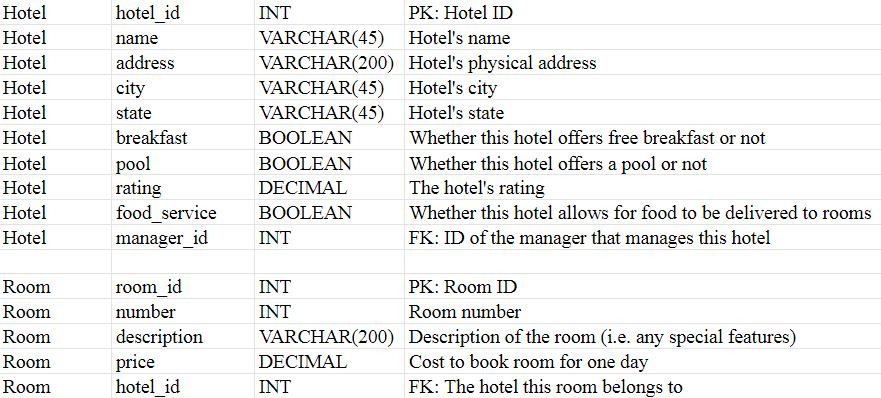
Customer will be sending requests to the database via GUI. The program will allow the clients to choose the hotel, dates, and provide optional feedback. The process should be quick-working and intuitive.

* + Manager

The manager will mostly be a series of setters and getters that refer to the database. This will be used by the manager of a given hotel, and be used by them to adjust prices and add rooms to the database.

* System (Tables and Description)
  + Data analysis
    - Data dictionary





* Algorithm Analysis
  + Big - O analysis of overall System and Subsystems
    - GUI

The big O analysis for the GUI is O(1), since it is constant, once the user logs in, it will direct them to either the employee or the customer subsystems

* + - Database

The time analysis of querying data in the database is O(log(n)).

* + - Customer

Big O Analysis: O(log(n))

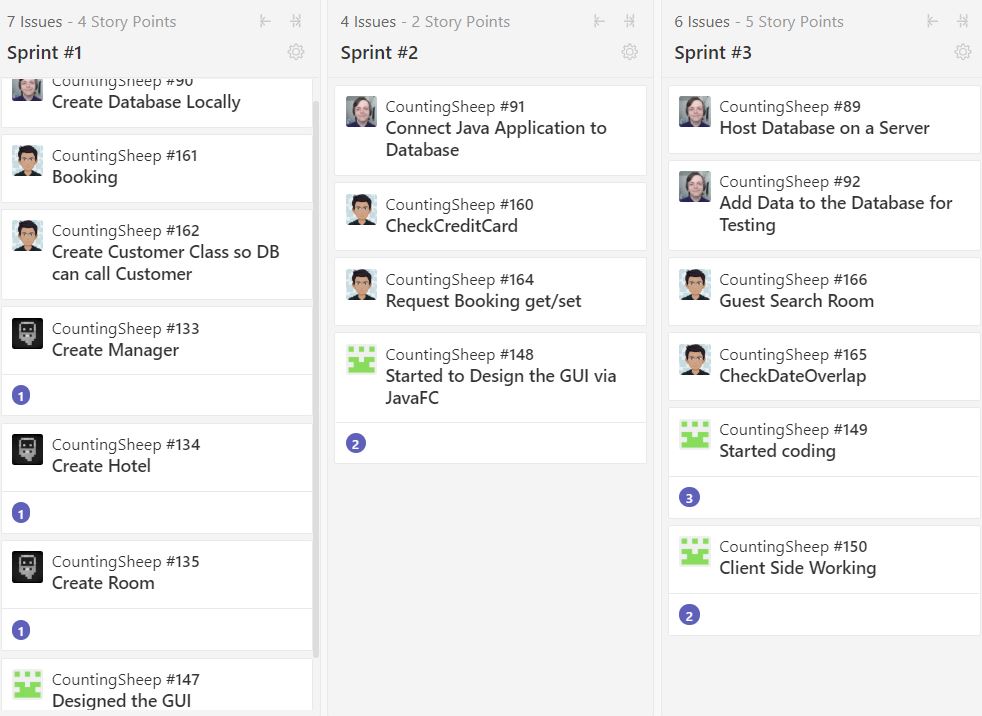
* + - Manager

The big O analysis of the manager algorithms will be O(log(n)) since it will rely on the way the database is constructed.

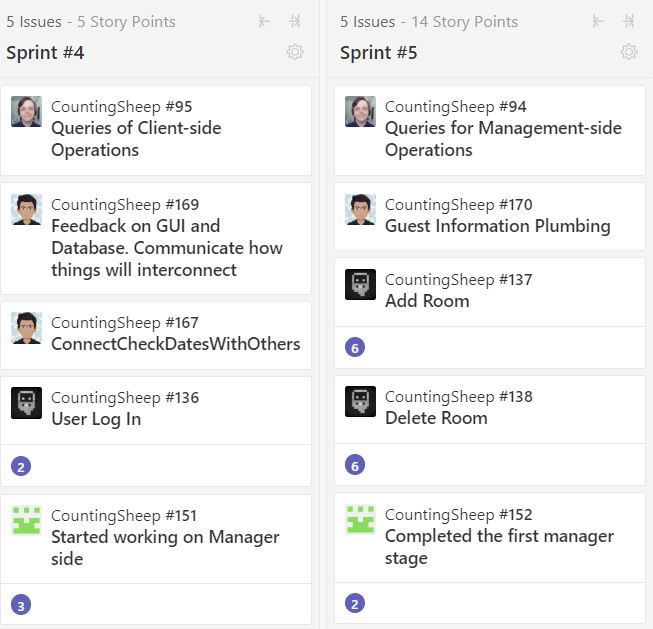
**6. Project Scrum Report -** *Group Responsibility*

* Product Backlog (Table / Diagram)
* Sprint Backlog (Table / Diagram)

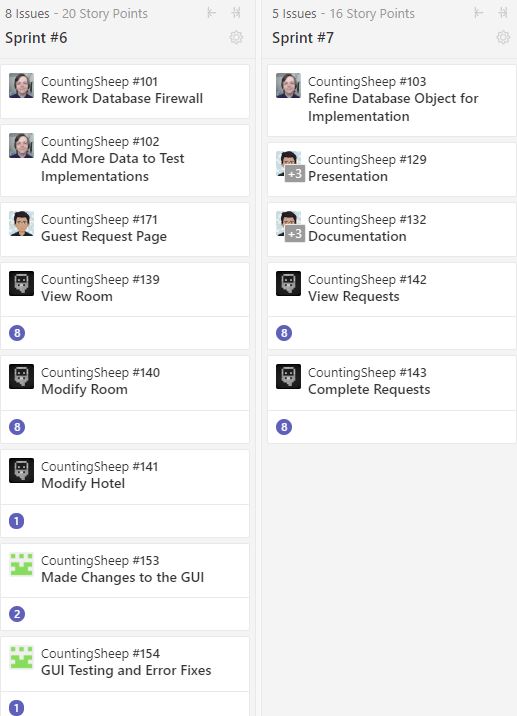
Sprints 1 through 3 (2/14/19 - 3/14/19)



Sprints 4 and 5 (3/14/19-3/28/19)

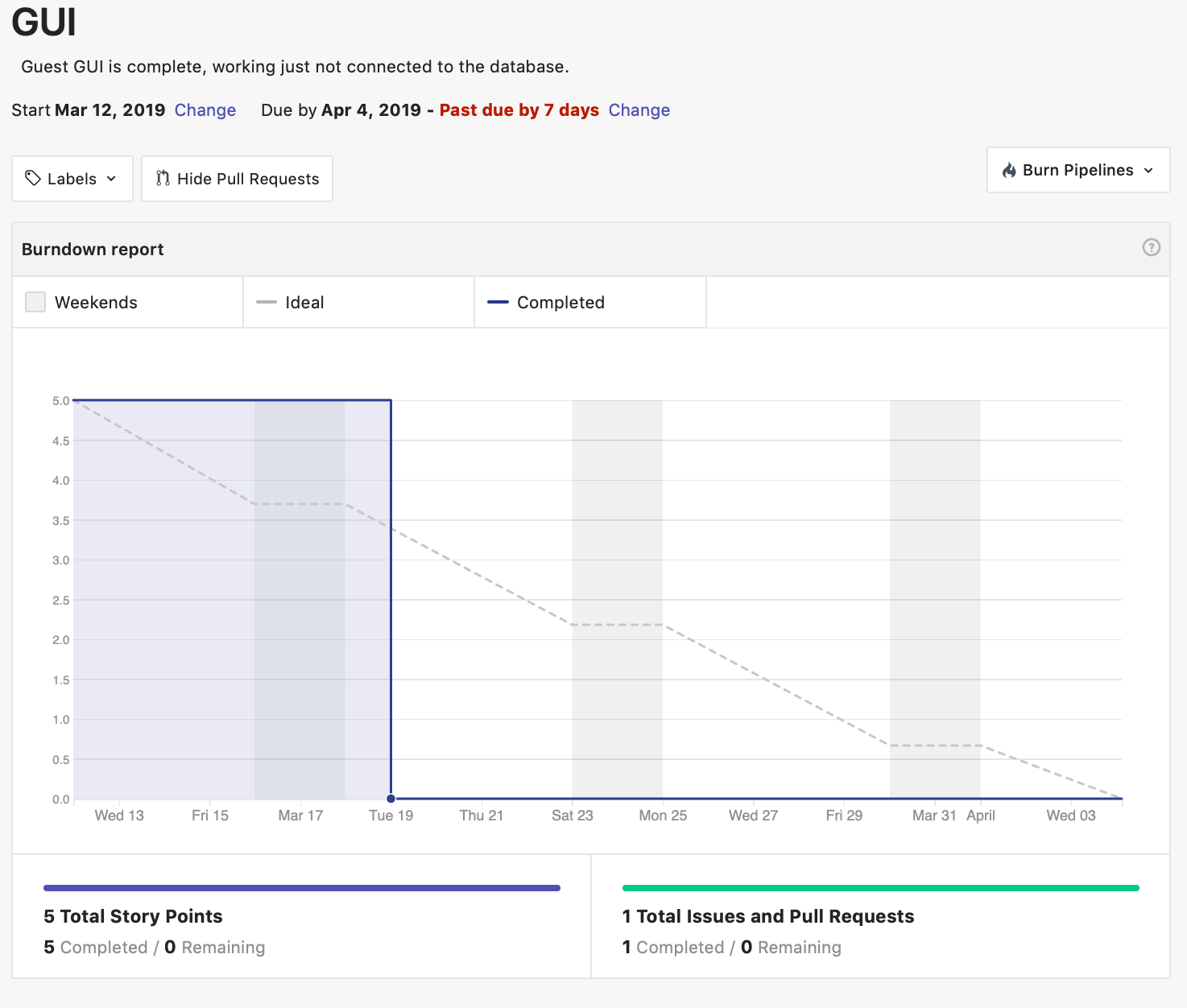


Sprint 6 and 7 (3/28/19-4/11/19)

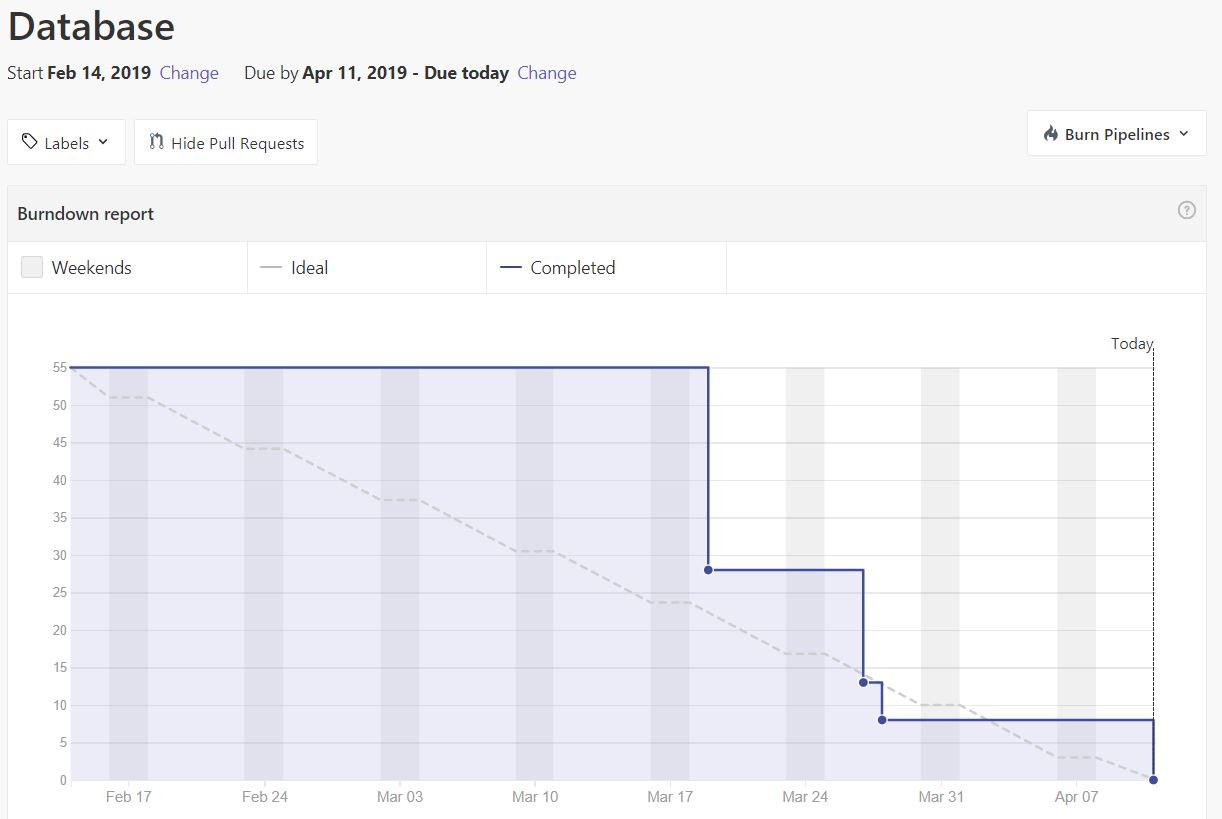


* Burndown Chart

GUI



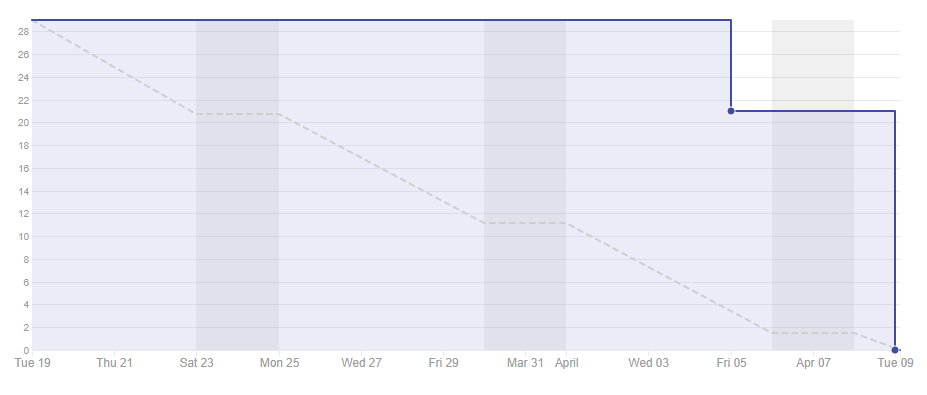
The burndown chart for the database subsystem:



Guest subsystem:

* 

Manager subsystem:



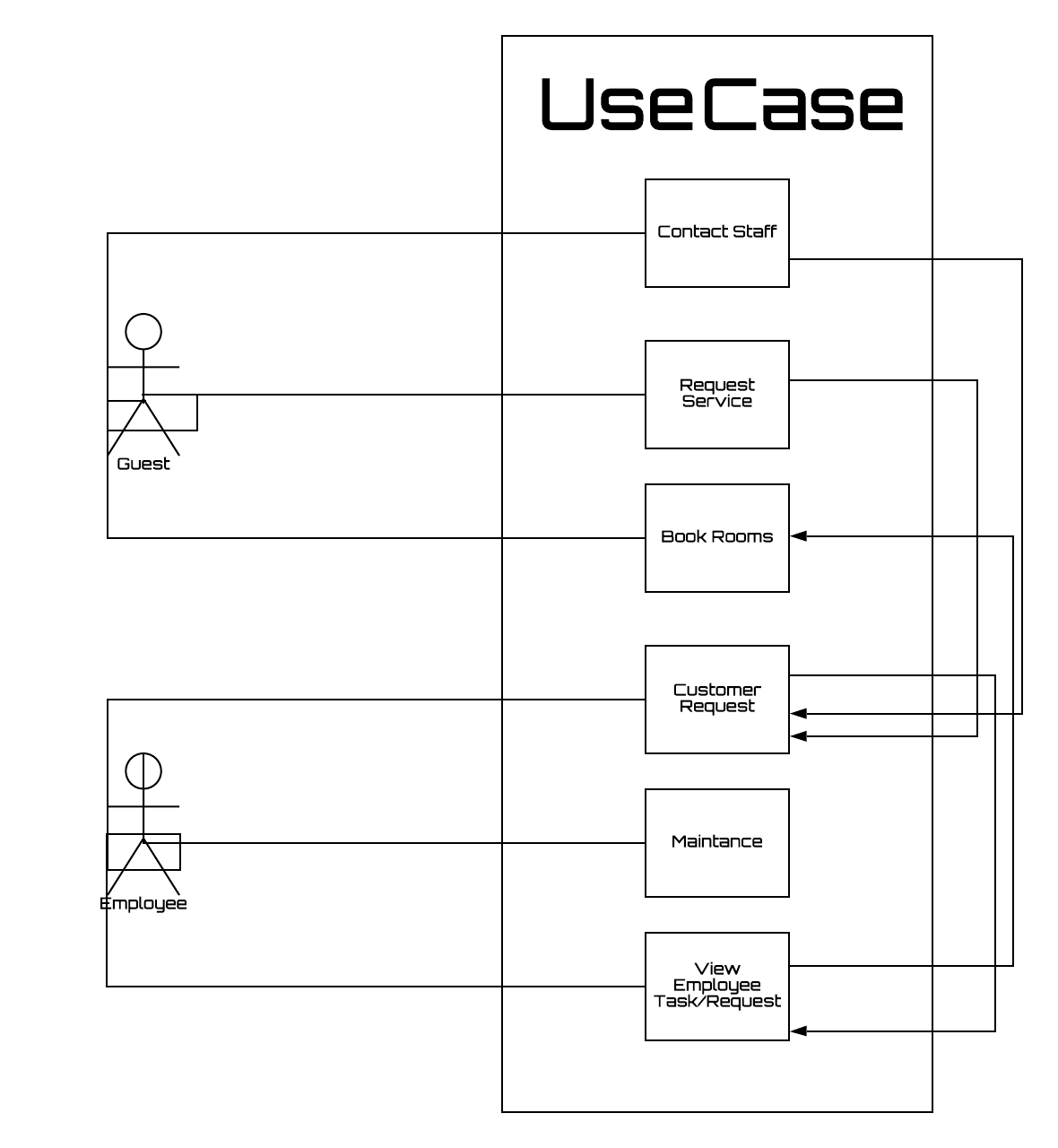
**7. Subsystems**

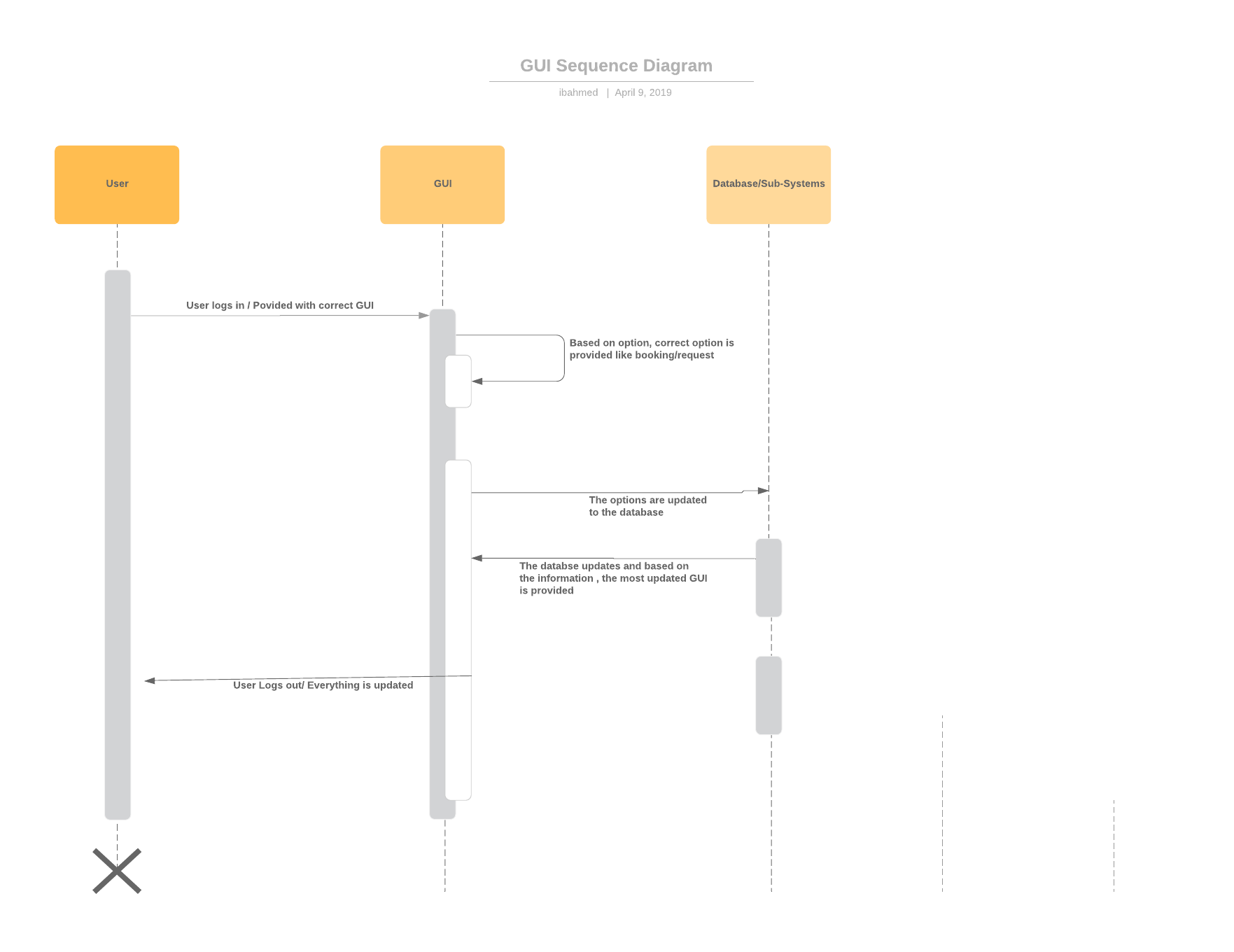
**7.1 Subsystem 1** – Name 1 - GUI

Initial design and model

The Initial design and model has changed quite a bit. We changed the way the request system works with connecting both the client and manager side. When testing throughout the entire process, we saw better ways to make both sides work, and changed them up. For instance, we changed the way the requests are made and how they are shown in both the client and manager side. The same went when we were showing rooms available for booking, we make everything into a big scroll bar which made everything easy to see.

* + Illustrate with class, use-case, UML, sequence ..... diagrams
  + Design choices
    - We chose the GUI in the way since it was the smoothest way to connect both the client and manager together. The purpose of this project was to make an application where multiple hotels and clients can all use the same application and be connected and not have to use a lot of different sites which can get confusing. So any manager and client can all use the same application, make requests, book rooms, complete request and from one application.
* Use Case Diagram

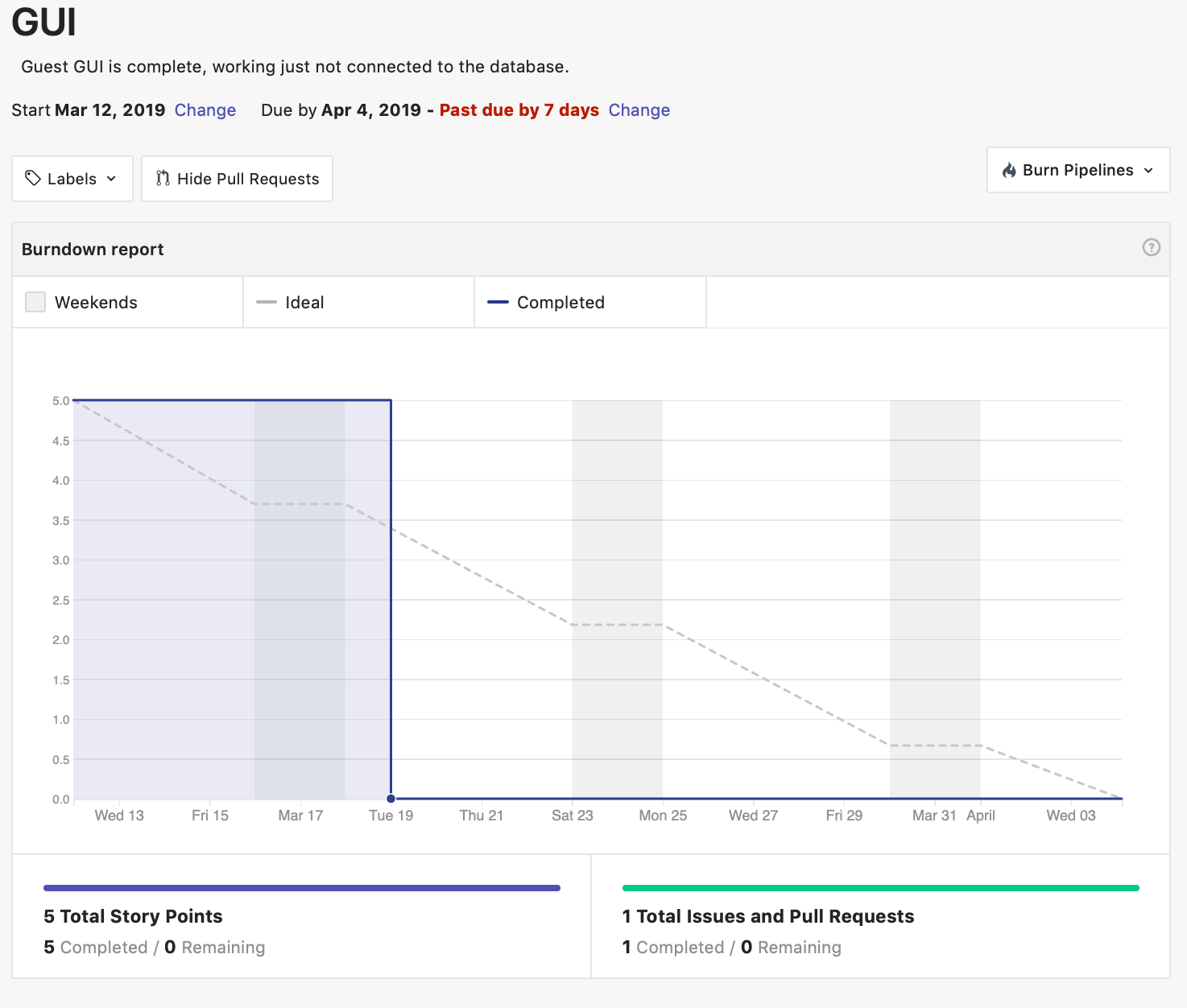




* If refined (changed over the course of project)
  + Reason for refinement (Pro versus Con)
    - Just changed some layout of the project. We did not make any major changes to the GUI.
  + Changes from initial model
    - We only changed some layouts, created scroll bars instead of making everything available in different pages.
* Scrum Backlog (Product and Sprint - Link to Section 6)
* Coding
  + The Application was coded using JavaFX on netbeans and eclipse. The reason for javaFX was that we had decided to code the entire project in java and the best way to connect everything together with the least difficulties would be trying to code everything in the same language.
  + Approach (Functional, OOP)
  + Language
    - The GUI was designed using JavaFX. The coding environment was eclipse and net beans.
* User training
  + Training / User manual (needed for final report)
    - The user logs in and makes request
    - The Manager logs in and completes request
    - Need to use correct login information
* Testing

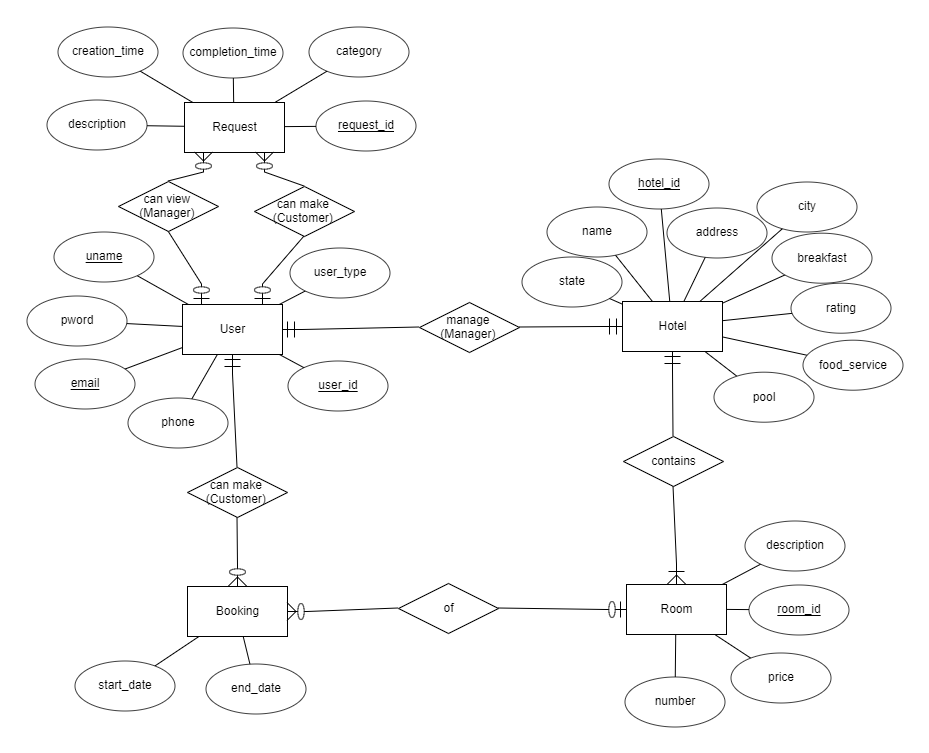
When testing the application, we would test from all different computers and different users to see if both sides get updated at the same time. For instance if client 1 books a room and hotel 1, it would show that he is booked there. When the client makes a request, it would show on hotel 1 side and the hotel would complete it and the guest would see that it is complete on his side.

Burn Down Chart

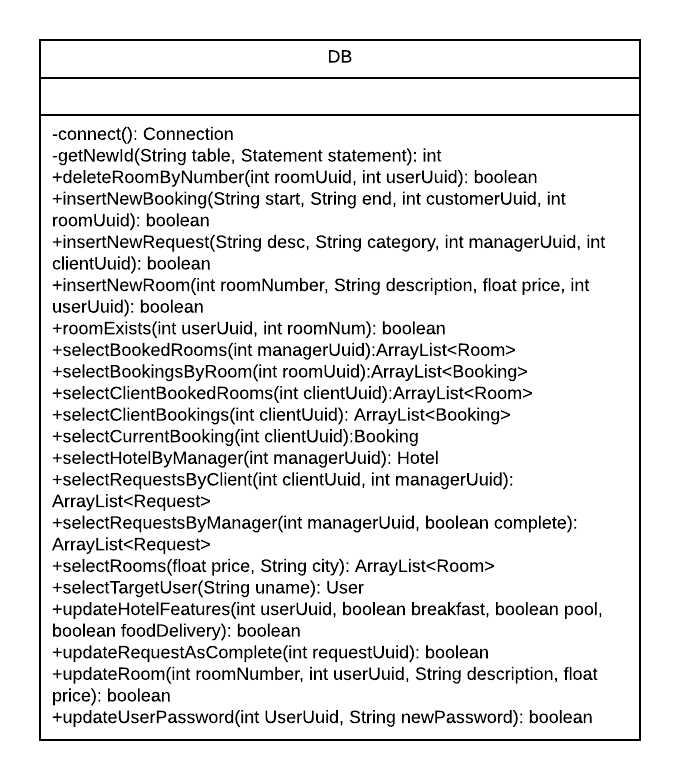


**7.2 Subsystem 2** – Walt Brady - *Database*

* Initial design and model
  + Illustrate with class, use-case, UML, sequence ..... diagrams



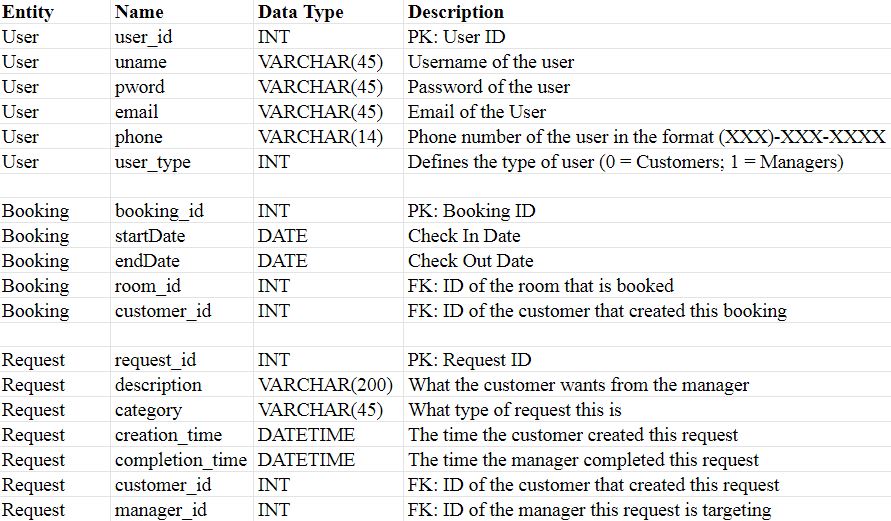
ER Diagram showing the final design of the database

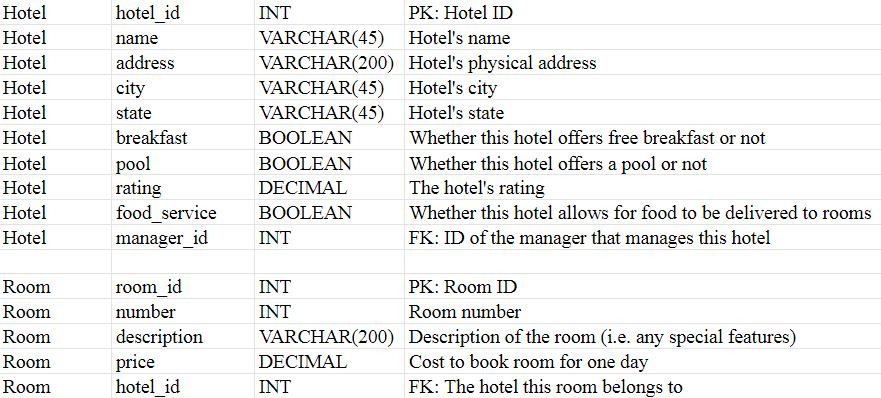
UML of the database object used by the Java application in order to access the mySQL database.

* + Design choices

This subsystem consists of the mySQL database and a Java object in the application that handles all database needs. The reason why mySQL was chosen was to take advantage of the capabilities of a relational database, as the objects within a hotel management system often need to interact with one another. The Java object was a design choice that was allow the application to connect with the database quickly and safely as the object limits each user’s access to the overall database.

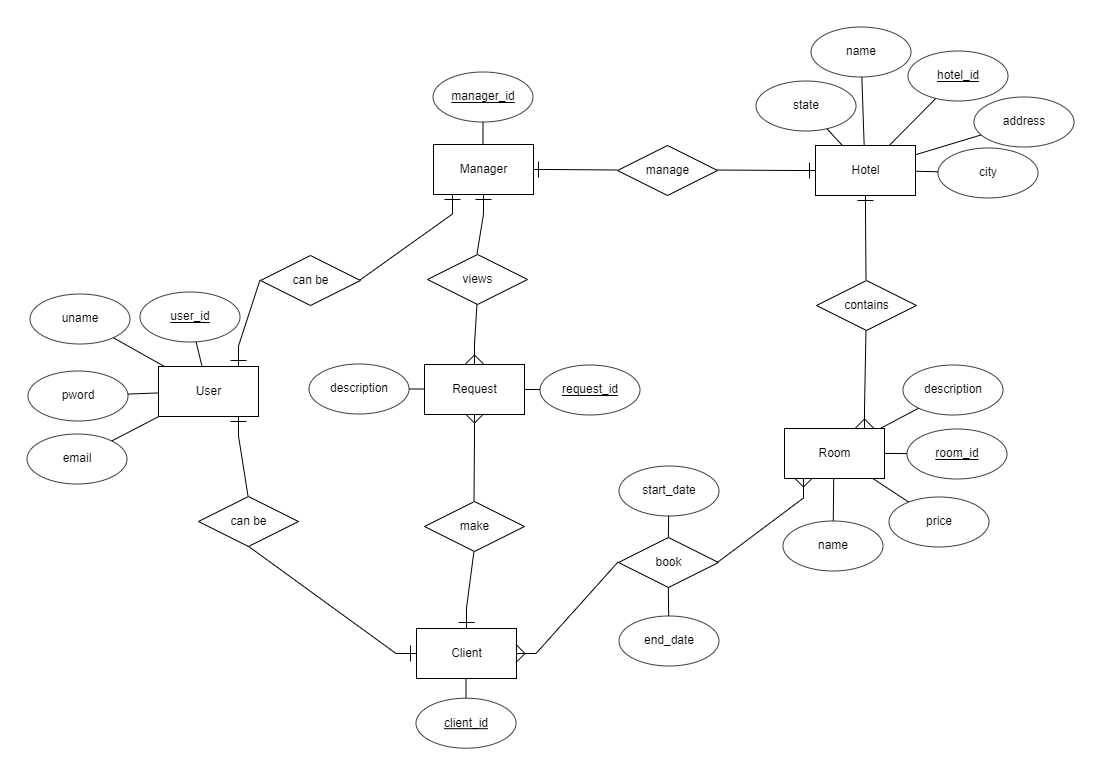
* Data dictionary





* If refined (changed over the course of project)
  + Reason for refinement (Pro versus Con)
  + Changes from initial model
  + Refined model analysis
  + Refined design (Diagram and Description)

The original model for the database included a couple of extra tables, called Manager and Client. The old ER model for the system is as follows:



This model would be good if there were any exclusive variables for either managers or clients. However, since both managers and clients both share attributes of user, indicated by the “can be” relation, and there does not exist any unique any unique attributes for either managers or clients, then the two tables can be replaced by an attribute in the User table, indicating whether the user is a client or a manager. The reason behind doing this change for the final model is for both simplicity and to improve computation time as it would reduce the search time in the database as there will be less tables to search through. One other change from the initial model to the final model is the inclusion of hotel features, those being whether they offer breakfast, a pool, or room service. This change was made to better reflect the scope of the project. One final change made to the initial model is labeling the “book” relation its own entity. The reasoning behind this change to the model is to better reflect the implementation of the actual database.

* Coding
  + Approach (Functional, OOP)

The database itself is made using Functional programming, however the database object within the application is designed as an object for other objects in the system to use.

* + Language

The database was created using mySQL and the database object was created using Java.

* User training
  + Training / User manual (needed for final report)

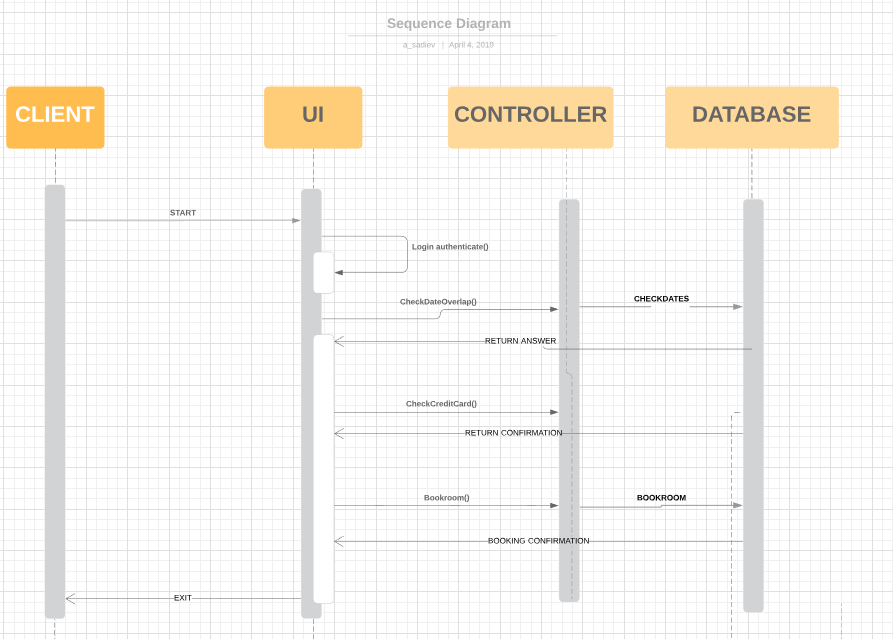
The database subsystem does not allow users to directly access the database. The database is instead accessed through other subsystems. For example the manager subsystem will request to see which rooms are in a manager’s hotel. This is to simplify database actions for the user and to limit what all the user can change in the database.

* Testing

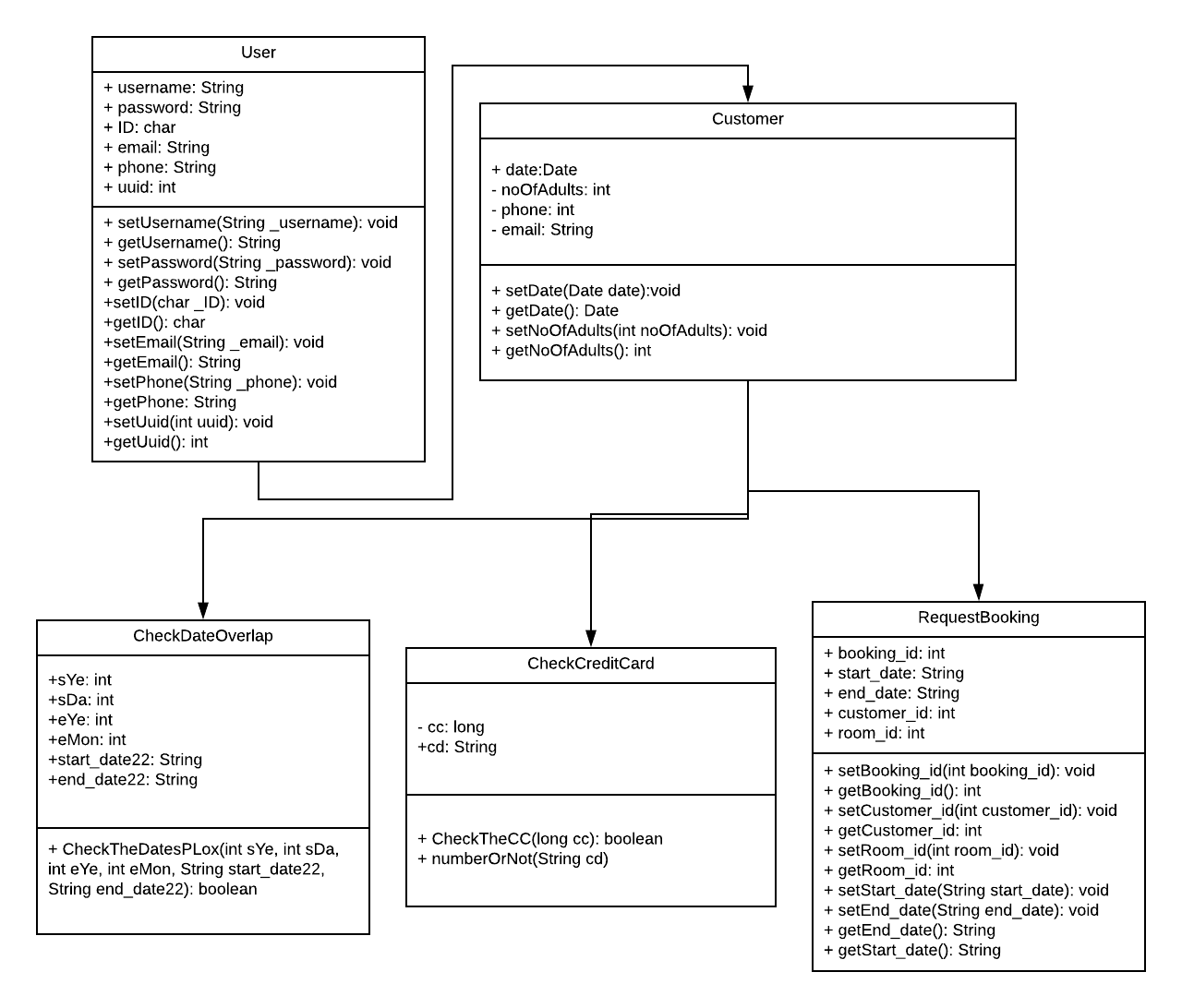
The queries in the database object were tested using mySQL workbench before being tested in the Java application.

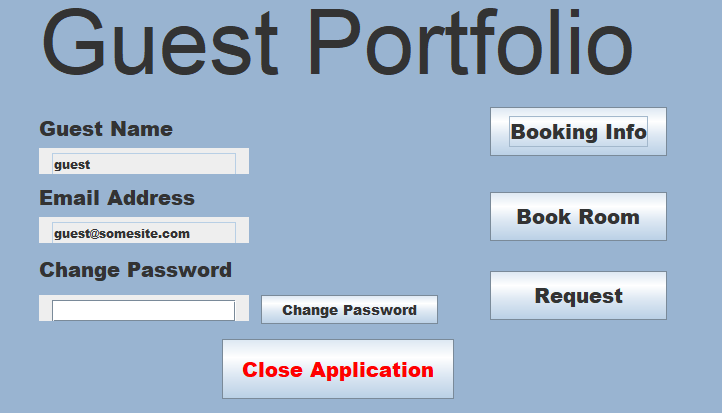
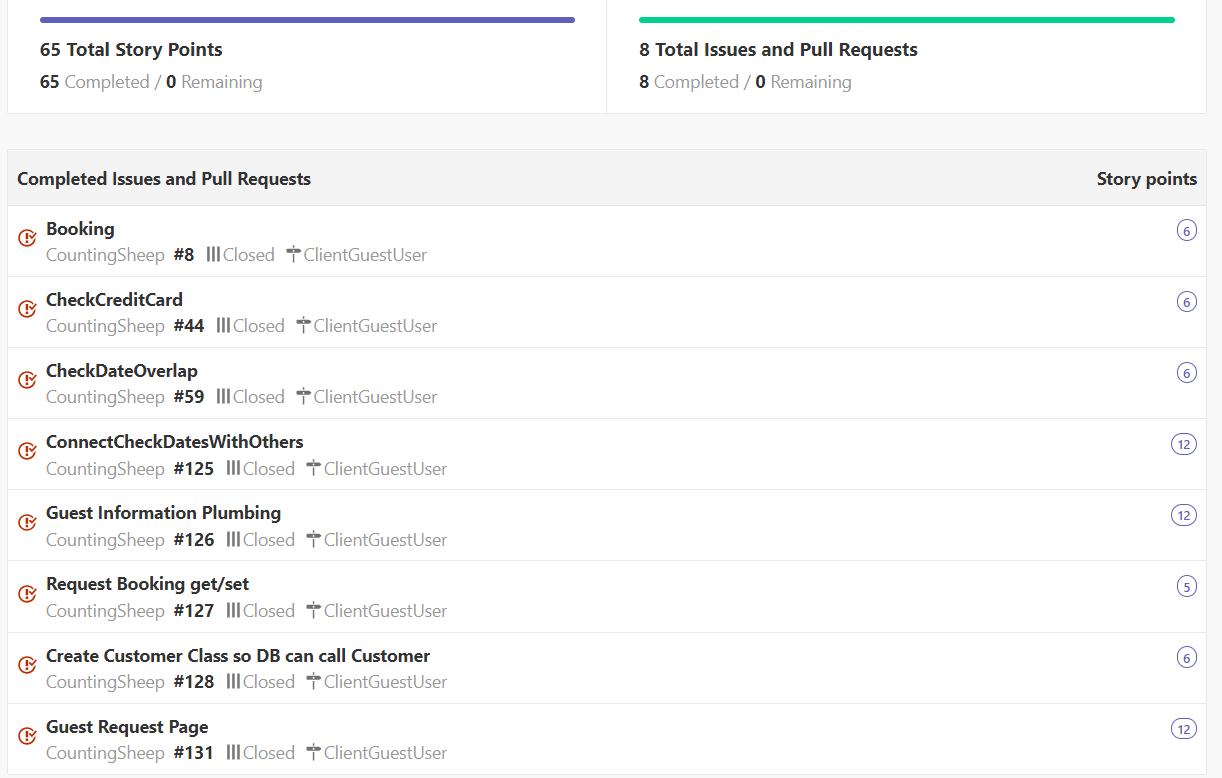
**7.3 Subsystem 3** – Name 3 - *Guest/User/Client*

* Initial design and model
* Sequence Diagram:



* UML Diagram:



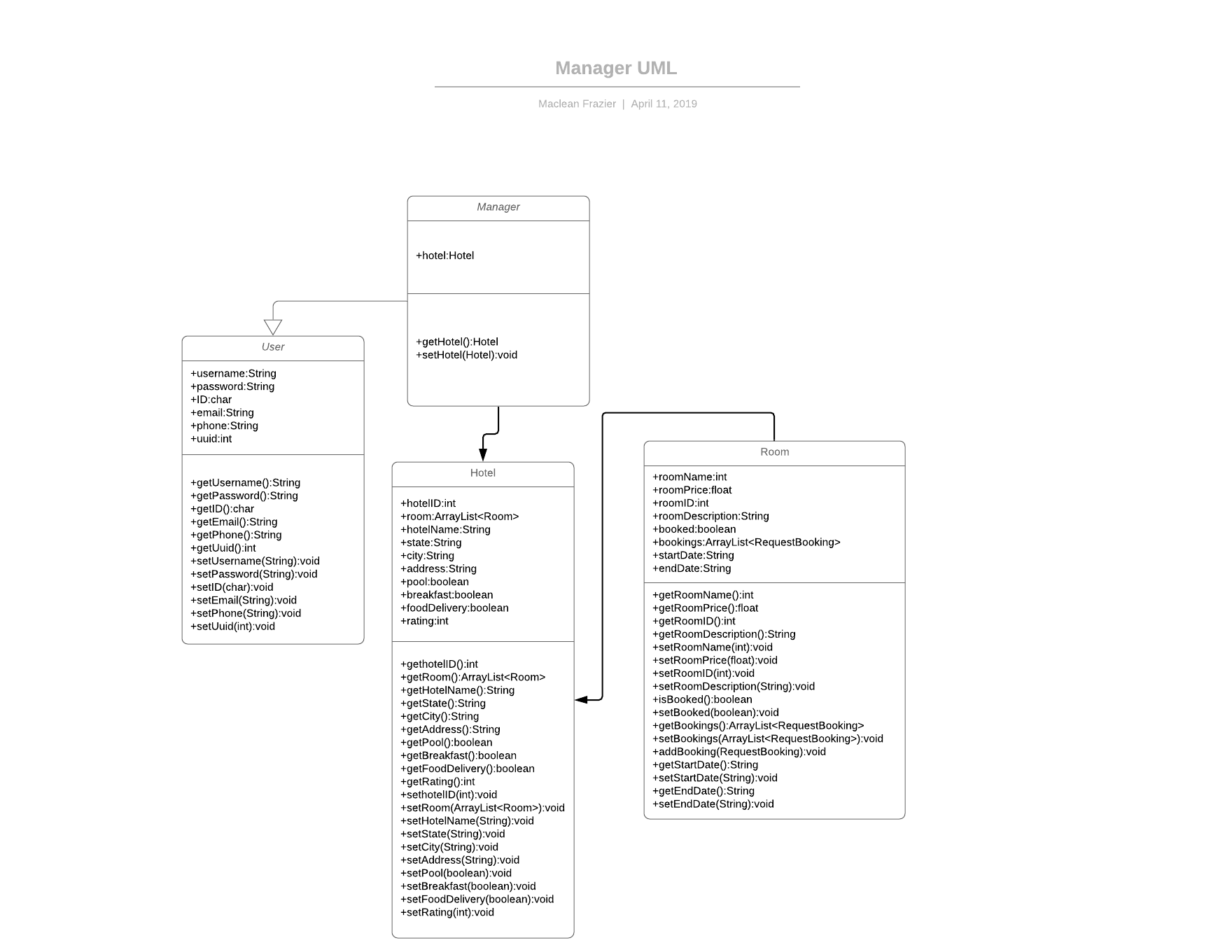
* DESIGN:
  + **Initial design** (we used the same initial design but added more features): Client side will use the UI to send requests such as choose date of bookings, list available hotels, as well as book rooms. These processes will send requests to the server for the respective information. Once processes are fully finished, the client will see a notification (confirmation) of completion. The whole project is based around the Java language, Java FX (GUI) and mySQL database.
  + **Things Changed:** We added more usability such as the ability to search for room by city/price, pick out a hotel from a list of available hotels, validate and check for user input errors, the listing user information, and a list a history of guest requests made. These changes were beneficial considering that it added more flexibility for the user.
* Sample screenshot of the Guest Menu: 
* Scrum Backlog (Product and Sprint - Link to Section 6)  
  
* Coding
  + We implemented Java’s OOP capabilities such as class inheritance, abstraction, polymorphism and encapsulation.
* User training
  + Training / User manual (needed for final report) general information for non devs (general population) How to use it/Description

The user will log in as a “guest” with their password which is also “guest”, upon correct user input the “Guest Portfolio” is opened. Guest Portfolio will have the guest name and email displayed. They will have the option to change their password. While still on the Guest Portfolio menu, the user may choose one of the following three choices. The first choice is “Booking Info” to look at their booking information such as hotel name, room number, room price, features and dates of booking. The second choice the user may click is “Book Room”, which will help the user search for a room by city, price and dates. Upon choosing a valid city that have available hotels based on the user's parameters, the user will be displayed a choice of hotels to choose from. The third and final choice from Guest Portfolio menu is the “Request” option, which allows the user to request food, maintenance service and/or leave feedback as well as see a history of request which will display if the requests were completed and uncompleted. All three options Booking Info, Book Room, and Request will have the option for the user to navigate back to the main menu of “Guest Portfolio”.

* Testing: We tested our project via GitHub with every change we made to make sure everything worked and that everyone was on the same page regarding progress made.

**7.4 Subsystem 4** – Name 4 - *Manager*

* Initial design and model
  + Illustrate with class, use-case, UML, sequence ..... diagrams
  + UML Diagram:



* + Design choices
* If refined (changed over the course of project)
  + Manager was changed from a separate object to an extension of User, which eliminates the need for manager username and password variables.
  + Pool, food delivery, and breakfast boolean values were added to the hotel so that they can be expressed in the database, and changed in the program itself.
  + We decided to express rooms in a hotel as an array list of room objects attached to the hotel. This lets us quickly cycle through the rooms in a hotel in an ordered manner. The same is true for the booking object in Room.
  + Booked is added as a boolean to room so that we can easily check the status of a room.
* Coding
  + I used Java, as well as helping modify the GUI using JavaFX. I used an OOP approach to this project.
* User training
  + Training / User manual (needed for final report)
    - User will log in as a “manager” tagged user, meaning they will have access to the manager systems but not the guest.
    - User will then have three options from there. Room Modification, Request, Hotel Options, as well as a button to close the application.
      * Room Modification opens three other menus, Add Room, Delete Room, and Modify/View Room, as well as a button to go back to the previous menu.
        + Add Room will open a menu that prompts the user to enter a room number, room description and room cost. If the room number entered is already taken, then a menu will pop up warning the user of that. Once they enter the desired information the user will hit submit, and be taken to the previous menu. The user can also go back via the “Previous Menu” button.
        + Delete Room will open a menu prompting the user to enter a room number. If the room number doesn’t exist, the user will be told such. Otherwise the room with the appropriate room number will be deleted from the database, and the user will go back to the previous menu. They can also go back via the “Previous Menu” button.
        + Modify/View Room will bring up another menu where the user will be given the option to View Room or Modify Room, as well as going back to the previous menu.

View Room brings up a scrollable text area listing the rooms currently under the user’s hotel. There is a button to go back to the previous menu.

Modify Room will bring up another menu with three text fields the user can type in. Room Number will be the number of the room the user wishes to modify. If the number doesn’t exist, the user will be prompted to enter a new number. The other two fields will be the user’s new price and description of the room. Once they hit “Update” the user will be taken to the previous screen. The same will happen if they press the “Previous Menu” button.

* + - * Request will take the user to a seperate menu with a “Complete” and “Incomplete” menu options, as well as a button to go back to the previous menu.
        + Complete will bring up a scrollable text area that lists the completed requests attached to the user’s hotel. “Main Menu” will take the user to the previous menu.
        + Incomplete will bring up a scrollable text area that lists the incomplete requests. Each request has a unique request id that the user can enter to mark the request as complete. “Main Menu” will take the user to the previous menu.
      * Hotel Options brings up a seperate menu with three checkboxes listed “Breakfast”,“Pool”, and “Food Delivery”. The boxes checked will trigger the appropriate boolean values on the hotel object to trigger. Unchecked boxes will change those values to false.

**8. Complete System** – *Group responsibility*

* Final software/hardware product
* Virtual Java Application Executable
* Source code and user manual – screenshots as needed - Technical report

<https://github.com/amir111/CountingSheep>

* Evaluation by client and instructor
* Team Member Descriptions
  + Walt Brady- Database
  + Imran Ahmed- GUI
  + Maclean Frazier- Manager
  + Amir Sadiev- Client