Bilkent University

Department of Computer Engineering

Senior Design Project

whotello: time to interact with your hotel

Low-Level Design Report

Alina Zhumasheva, Bayram Muradov, Imran Hajiyev

Supervisor: Uğur Doğrusöz
Jury Members: H. Altay Güvenir, İbrahim Körpeoğlu

Low-Level Design Report
June 24, 2019

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Senior Design Project course CS491/2.

Table of Contents

1 Introduction

The number of hotel users around the globe is quite large, surpassing the benchmark of 15.5 million rooms available for active usage scattered around the globe [1]. However, despite the fact that hotel service exists from 705, even nowadays using hotel services sometimes becomes challenging for both the guests and the administrators [2]. Reserving restaurants, chaise longues, requesting the cleaning or food, learning answers to frequently asked questions and even managing the room temperature for guests; managing the personnel, user preferences, food habits, overall hotel statistics for administrators may present itself a bit difficult due to the human factor in all of the processes and all of them may be optimized for the both groups benefits.

Our system is designed in a way to tackle and solve all of those problems, making the guest experience at the hotel as effortless and pleasant as possible and at the same time easing and simplifying the managing process for the administrators, giving the latter undesired benefits such as specified statistics and data for the further improvement of the hotel service - and all of this is automated, removing the human factor.

1.1 Object Design Trade-offs

1.1.1. Functionality vs. Usability

Functionality and usability are one of the most important parameters for the “whotello” project. Since the main purpose of the project whotello is to make the hotel industry as easy and simple to use as possible - helping both Guest and Administration side to do their main tasks in a simplified and comfortable way - only the core and must-have functionality will be included. If the project will carry out a lot of functionality, then automatically it becomes much harder to use. Therefore, it is important to decide which features are decreasing the usability parameter and which ones we can implement while maintaining a good UX and simple UI.

1.1.2. Cost vs Availability

Connecting to the room devices, such as lights, TV, AC, can be done in 2 ways. It is possible to connect to modern devices using special protocols, without the usage of Arduino, or Rasberry Pi. The cost will be low, as no additional devices have to be bought and placed in each hotel room. However, that comes with a trade-off of availability. Old models, won’t be able to be controlled by the Guest’s phone, because they can’t detect protocols that modern models can. Therefore, for old models, it is a must to buy and place either Arduino or a Rasberry Pi, to be able to

control them using a mobile phone. However, that results in an increase in the cost. However, the more devices our project can control, the better, as it will result in high availability.

1.2. Interface Documentation

Unified Modeling Language (UML) diagrams were used during documenting the low-level interface of the “whotello” project, in order to show the classes, variables, functions and their relationships with each other [3]. In UML representation, the class name is written on the top of the box. Under the class name, there might be variables that belong to the class. The visibility of a variable is shown by using +, -, # and - signs. After class variables are given, functions that belong to the class are listed. In that part, “functionName(parameterName: parameterType, ...)” format is used in order to represent function name, its parameters, and their types. An example class diagram that is following the interface described above can be found below. All other class diagrams provided in this report follows the same interface.

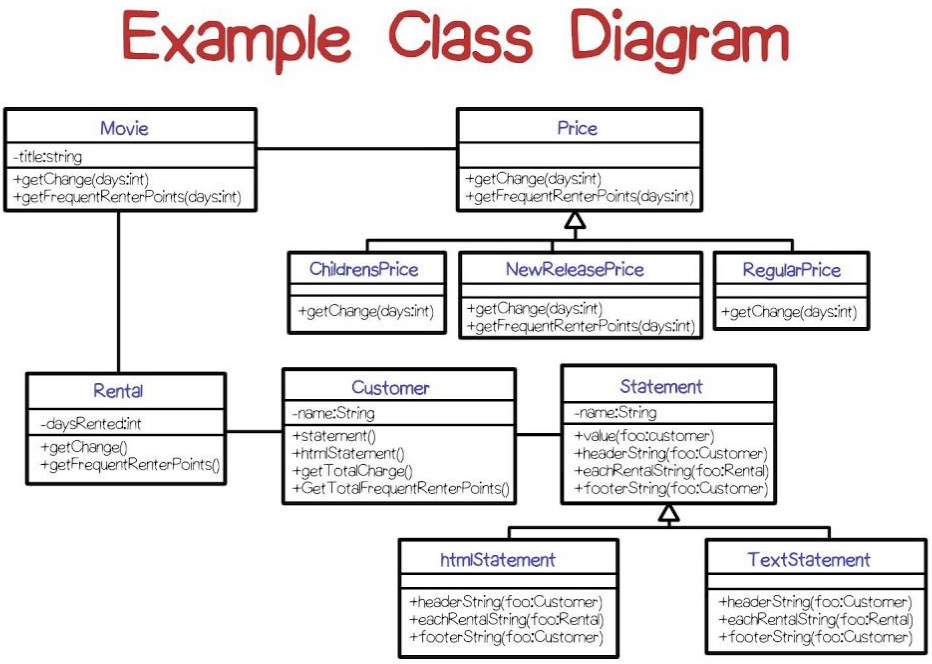


Figure 1. Example Class Diagram

1.3. Engineering Standards

To describe the classes, scenarios, use cases, subsystems, components, etc. We used UML design principles. We chose UML as it is the most common standard used to represent the structure of the system and its functionalities. The references

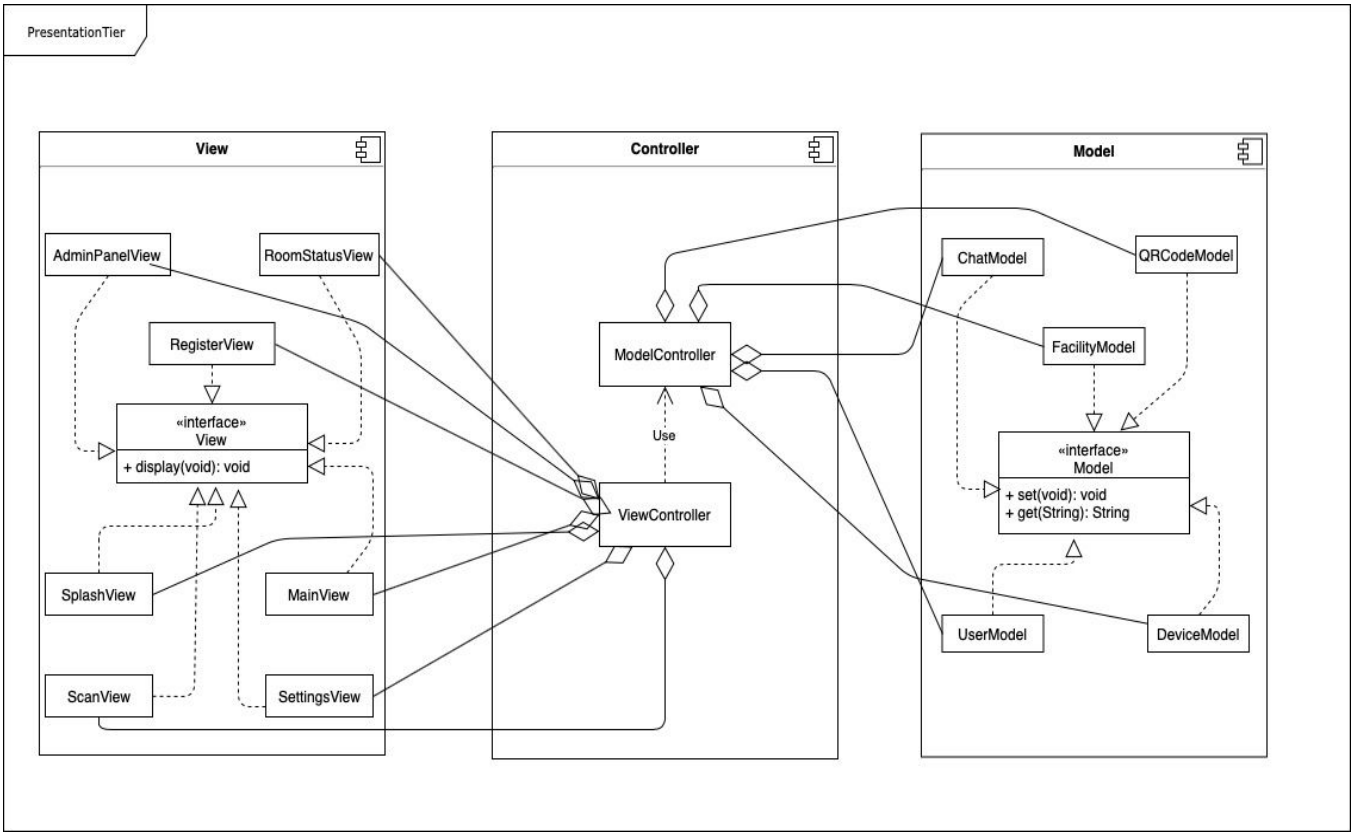
we provide are written in IEEE standard because it is very common and easy to read.

1.4 Definitions, Acronyms, and Abbreviations

- API: Application Programming Interface
- iOS and Android: Mobile operating systems
- HDM: Hotel Database Management
- UML: Unified Modeling Language

2 Packages

2.1 Presentation



2.1.1 Controller

- **ViewController:** Provides interface between UI and backend of the application.

- **ModelController:** Provides interface between UI and Models of the application. Furthermore, constructs, controls and updates the models that need to be send to Application tier.

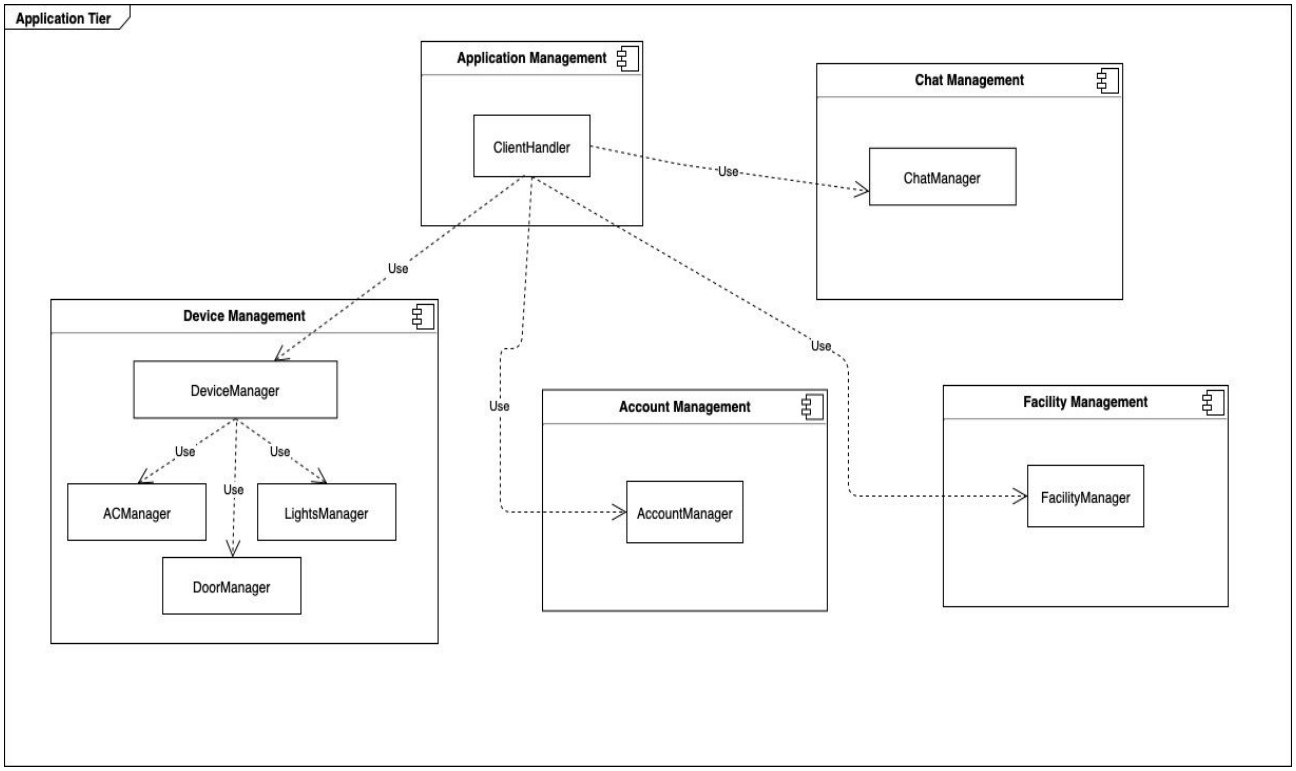
2.1.2 Model

- **ChatModel:** Keeps Chat data and updates them as needed.
- **FacilityModel:** Keeps Facility data and updates them as needed.
- **UserModel:** Keeps user data.
- **DeviceModel:** Keeps state of devices and updates them as needed.
- **QRCodeModel:** Keeps QR code related information.

2.1.3 View

- **AdminPanelView:** User Interface for administration panel.
- **SplashView:** User Interface for initial page of application.
- **RegisterView:** User Interface for registration page of application.
- **SettingsView:** User Interface for Device settings page of application.
- **ScanView:** User Interface for code scanning page of application.
- **MainView:** User Interface for main page of application.
- **RoomStatusView**: User Interface for displaying current state of devices in the room.

2.2 Application



2.1.1 Application Management

- **ClientHandler:** Processes main operations related with the subsystem.

2.1.2 Chat Management

- **ChatManager:** Processes chat requests, returns appropriate responses.

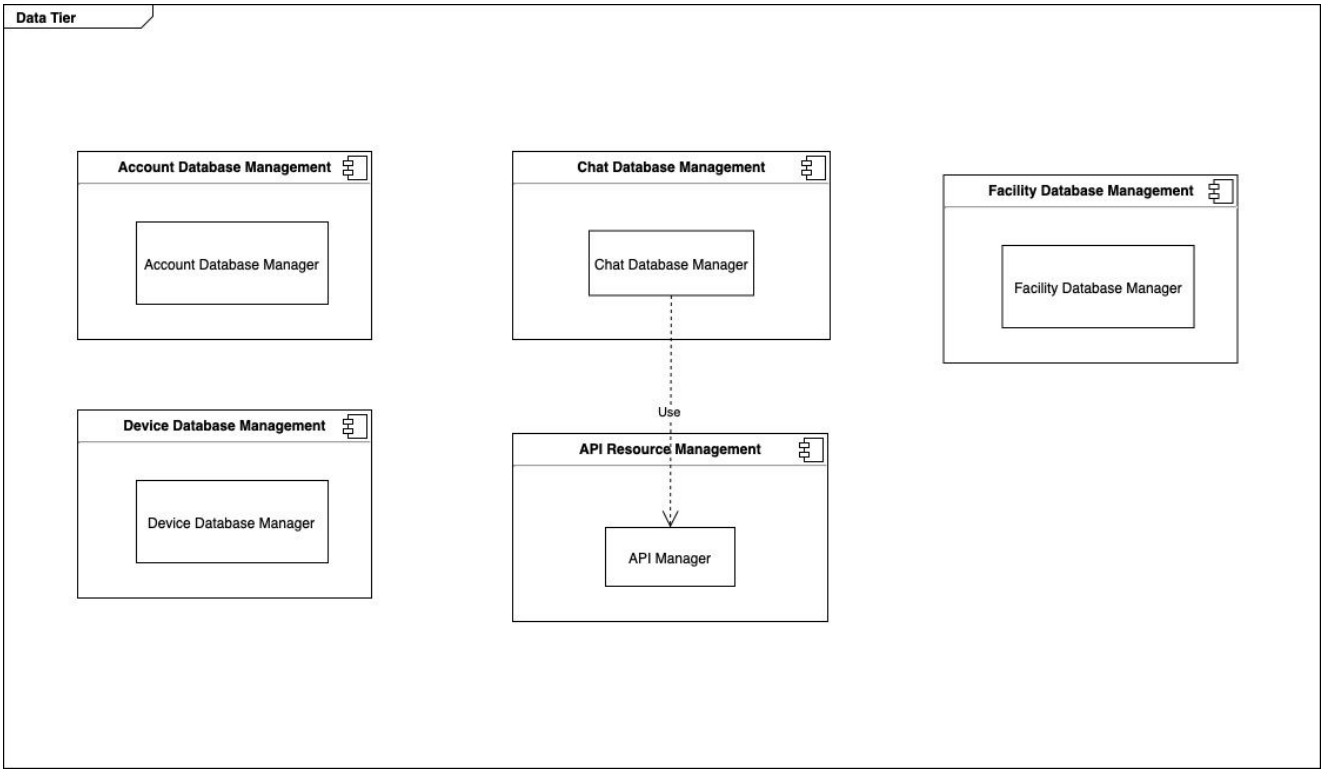
2.1.3 Device Management

- **DeviceManager:** Processes main operation related with Device Management.
- ACManager: Controls Air Conditioner.
- LightsManager: Controls Lights.
- DoorManager: Controls Door.

2.1.3 Account Management

- **Account Manager:** Processes operations related with user accounts.

2.3 Data Tier



2.3.1 Account Database Management

- **Account Database Manager:** Stores and fetches account related data.

2.3.2 Device Database Management

- **Device Database Manager:** Stores and fetches device related data.

2.3.3 Chat Database Management

- **Chat Database Manager:** Stores and fetches chat related data.

2.3.4 Facility Database Management

- **Chat Database Manager:** Stores and fetches chat related data.

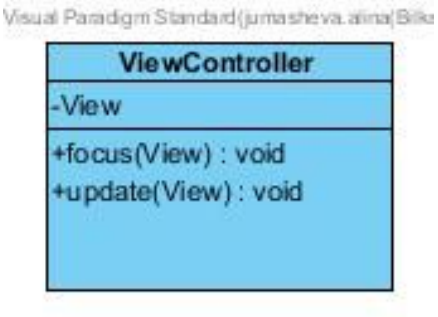
2.3.4 API Resource Management

- **API Manager:** Stores and fetches chatbot data by using various APIs.

3 Class Interfaces

3.1 Presentation Tier Interfaces

3.1.1 ViewController Class



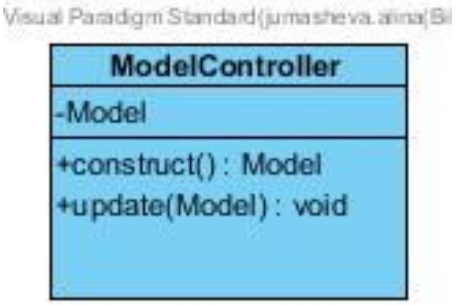
Attributes:

- **View:** holds reference to a specified View.

Methods:

- **focus(View):** focuses a specific view given in parameter.
- **update(View):** updates current view.

3.1.2 ModelController Class



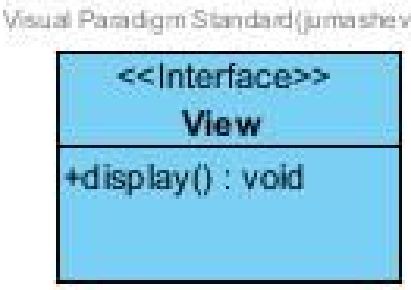
Attributes:

- **Model:** holds reference to a specified Model.

Methods:

- **construct(Model):** constructs a specific Model.
- **update(View):** updates a Model.

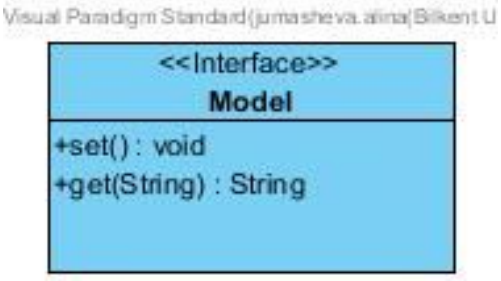
3.1.3 View Class



Methods:

- **display():** displays a View.

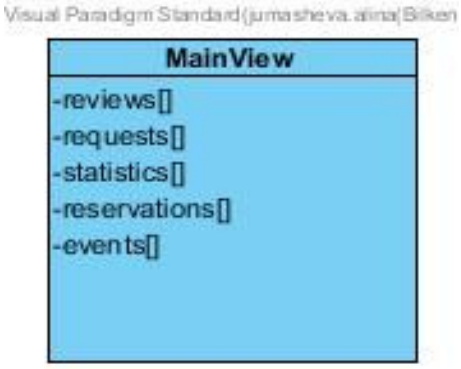
3.1.4 Model Class



Methods:

- **set():** updates an attribute.
- **get(String):** retrieves a specific attribute

3.1.5 MainView Class



Attributes:

- **reviews[]:** keeps user reviews about the hotel.
- **requests[]:** keeps requests made by user.
- **statistics[]:** keeps statistics about hotel.
- **reservations[]:** keeps reservations made by user.
- **events[]:** keeps list of events in the hotel.

3.1.6 RoomStatusView Class

Attributes:

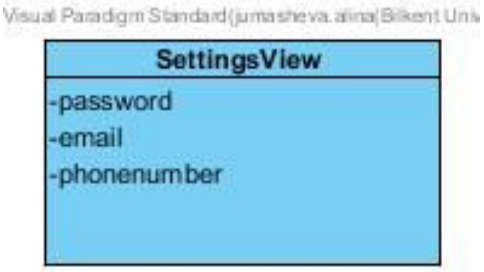
- **temperature:** keeps state of temperature.
- **light:** keeps state of lights.
- **door:** keeps state of door.

3.1.7 RegisterView Class

Attributes:

- **name:** keeps name input.
- **email:** keeps email input.
- **phonenumber:** keeps phone number input.

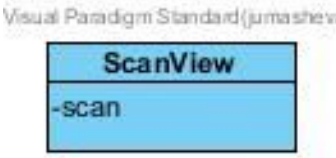
3.1.8 SettingsView Class



Attributes:

- **password:** keeps password input.
- **email:** keeps email input.
- **phonenumber:** keeps phone number input.

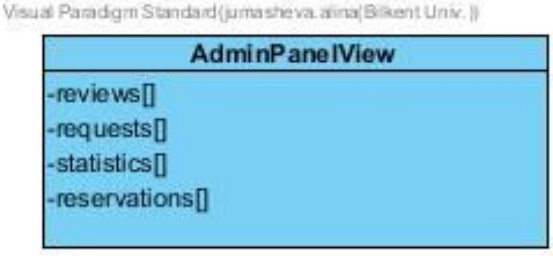
3.1.9 ScanView Class



Attributes:

- **scan:** keeps code scanner.

3.1.10 AdminPanelView Class

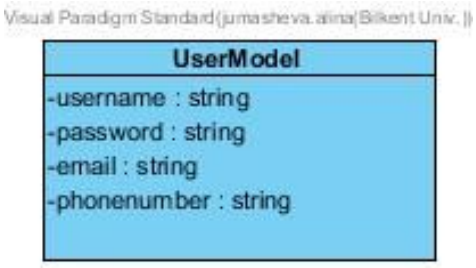


Attributes:

- **reviews[]:** keeps user reviews about the hotel.
- **requests[]:** keeps requests made by users.
- **statistics[]:** keeps statistics about hotel.

- **reservations[]:** keeps reservations made by users.

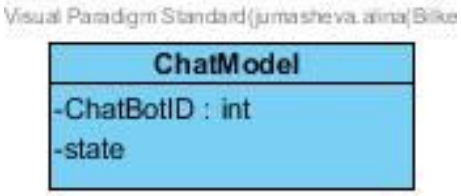
3.1.11 UserModel Class



Attributes:

- **username:** keeps username info of the user.
- **password:** keeps password info.
- **email:** keeps email info.
- **phonenumber:** keeps phone number info.

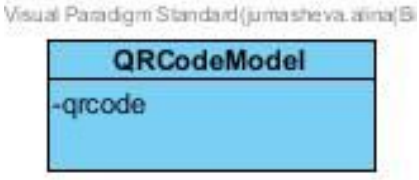
3.1.12 ChatModel Class



Attributes:

- **ChatBotID:** keeps id number of the chat.
- **state:** keeps state of the chat.

3.1.13 QRCodeModel Class



Attributes:

- **qrcode:** keeps generated qrcode.

3.1.14 DeviceModel Class

Attributes:

- **status:** keeps the status of the device.

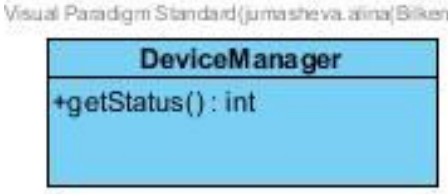
3.1.15 FacilityModel Class

Attributes:

- **arrangement:** keeps the arrangement info of the facility.
- **totalCapacity:** keeps information of total capacity of the facility.

3.2 Application Tier Interfaces

3.2.1 DeviceManager Class



Methods:

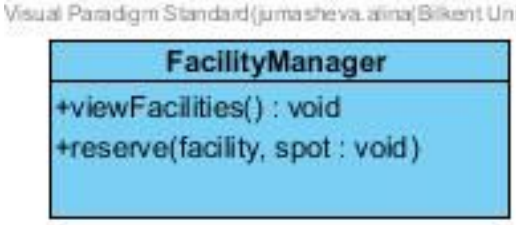
- **getStatus():** returns the status info of the device.

3.2.2 AccountManager Class

Methods:

- **createAccount():** creates and returns user account..
- **editAccount:** edits user account.
- **login(username, password):** logins using provided parameters(username and password).
- **authenticate:** authenticates user account.
- **logout:** logs out user account.

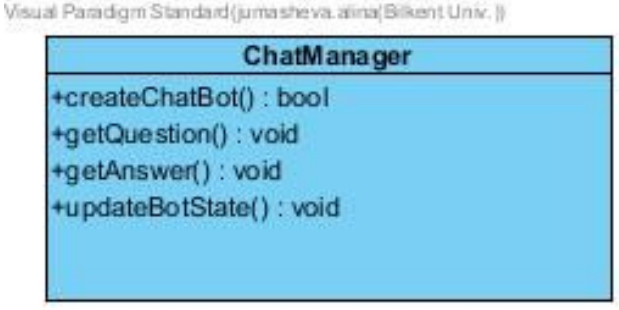
3.2.3 FacilityManager Class



Methods:

- **viewFacilities():** provides info about facilities.
- **reserve(facility, spot):** reserves spot at certain facility.

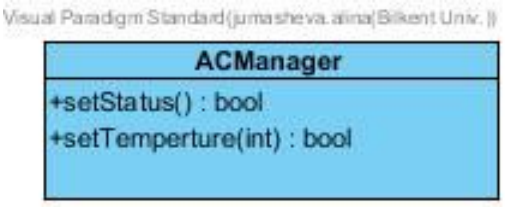
3.2.4 ChatManager Class



Methods:

- **createChatBot():** creates chatbot for a user.
- **getQuestion():** gets question from a user.
- **getAnswer():** provides answer for a question.
- **updateBotState():** updates state of the chatbot.

3.2.5 ACManager Class



Methods:

- **getStatus():** provides status of the AC.
- **setTemperature(int):** sets temperature for an AC.

3.2.6 LightsManager Class

Methods:

- **setStatus(int):** sets the status of the lights.

3.2.7 DoorManager Class

Methods:

- **setStatus(int):** sets the status of doors.

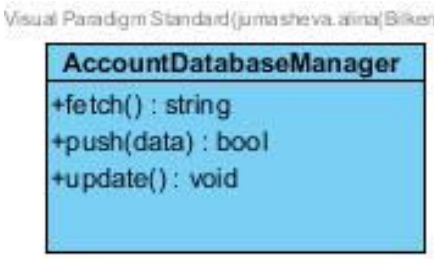
3.3 Data Tier Interfaces

3.3.1 DeviceDatabaseManager Class

Methods:

- **fetch():** fetches data from the device database..
- **push(data):** pushes data to the device database.
- **update():** updates data in the device database.

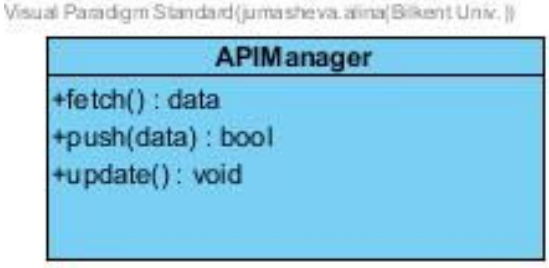
3.3.2 AccountDatabaseManager Class



Methods:

- **fetch():** fetches data from the account database..
- **push(data):** pushes data to the account database.
- **update():** updates data in the account database.

3.3.3 APIManager Class

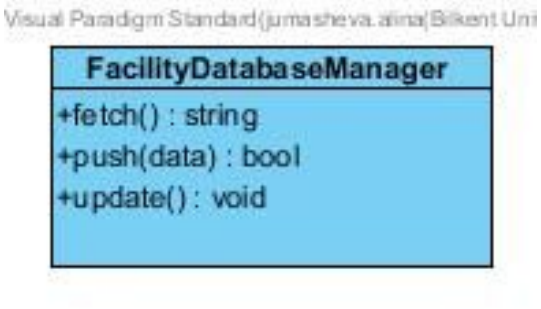


Methods:

- **fetch():** fetches data from the chat pool database.
- **push(data):** pushes data to the chat pool database.

- **update():** updates data in the chat pool database.

3.3.4 FacilityDatabaseManager Class



Methods:

- **fetch():** fetches data from the facility database..
- **push(data):** pushes data to the facility database.
- **update():** updates data in the facility database.

4 Glossary

Server is the system where access management to the service happens.

Client is the system where the user's interaction with the service happens.

Hotel Database Management is responsible for managing data related to the hotel.

5 References

1. Rebecca Lake, "*Hotel Industry Statistics*", 26-Apr-2016. [Online]. Retrieved from: https://www.creditdonkey.com/hotel-industry-statistics.html [Accessed: 24-Feb-2019].
2. Guinness World Records, "*Oldest Hotel*", 29-Oct-2011. [Online]. Retrieved from: http://www.guinnessworldrecords.com/world-records/oldest-hotel/ [Accessed: 24-Feb-2019].
3. en.wikipedia.org., *"Class diagram"* , 2019. [Online] Available at: https://en.wikipedia.org/wiki/Class_diagram [Accessed 18 Feb. 2019].