Design Document

Software Engineering Course Project

WASHETERIA

Team Number : 11
Team Members :

Sailendra D S (20161216)
Abhinav Anand (2018201037)
Nishant Goyal (2018201038)
Lokesh Singh Mahar (2018201049)
Neeraj Barthwal (2018201069)
Shubham Rawat (2018201098)

Table Of Contents

1. Overview

Problem Statement:

Develop a system to manage the usage of washing machines in a public washing machine setting typically a college hostel.

Motivation:

Congestion is often caused by students seeking the availability of a washing machine. Many a time queues of buckets(with clothes) throng the washing machine area because the students walk around to inspect the occupancy of individual machines.

Vision:

This project will provide an app which will show the current status of the washing machine(vacant/occupied), in case it is occupied, the app will show how much time is left before the machine is vacant. The app will also allow the end-users to reserve a slot based on availability in the near future.

2. Terminology

a. **Dashboard:** The initial screen shown to a user. This screen will show a list of available machines.
b. **Slot:** A time interval
c. **Reserve:** The process of booking a slot.
d. **Day view:** A client application's view showing time intervals for a chosen day.
e. **Assisted booking:** A mode of booking which automatically picks an available slot for the user.
f. **Live status:** The current timer being shown on the washing machines.

3. Tenets

a. **Functional Requirements**

Identifier	Requirement	Priority
REQ1	The system will authenticate the users using their IIIT H credentials.	Low
REQ2	The system will allow users to choose from a list of locations where washing machines are installed(OBH/NBH/Bakul/Parijat).	Low
REQ3	The system will allow users to view the live status of each machine(vacant/occupied/out of order).	High
REQ4	The system will show the current remaining time(of a wash) being displayed on each machine's control panel.	Medium
REQ5	The system will allow users to view free slots.	High
REQ6	The system will allow users to reserve a free slot.	High
REQ7	The system will allow users to cancel a previously reserved slot.	High
REQ8	The system will limit the maximum number of reservations that can be made by a user.	Med
REQ9	The system will limit the maximum duration of a slot.	Med
REQ10	The system will allow users to set reminders for a reserved slot.	Low
REQ11	The system will notify the user about 5min before its reserved slot and provide options(Going/Not Going). If a user selects Not Going then the reserved slot will become free.	Med

b. **Extended Functional Requirements**

Identifier	Requirement
REQ6a	The system will allow users to choose a date and time to reserve a slot.
REQ6b	The system will provide a *booking assistant* to users and automatically book a slot based on the user's preferences. (example: weekend, evening).
REQ8a	The system will limit the number of reservations per day to 3(configurable via backend).
REQ8b	The system will limit the number of reservations per week to 10(configurable via backend).
REQ10a	The system will remind the user 15 min before their reserved slot starts.
REQ10b	The system will remind the user 5 minutes before their reserved slot ends.
REQ10c	The system could allow users to set custom reminder time.

c. **Non-functional Requirements**

The system favors consistency over availability.

Identifier	Requirement
NFR1	If multiple users try to reserve the same slot at the same time exactly one user must succeed.
NFR2	If a majority of replicated servers are down, the system will allow only read operations and no write operations, i.e. users can view reserved slots but cannot make a reservation.

d. **Acceptance tests:**

Acceptance Tests for REQ1:
ATC1.01: Ensure user with valid IIIT-H email id to login and able to book a slot(Pass)
ATC1.02: Ensure user with any other email id fails to login and book slot(Fail)

Acceptance Test for REQ2:
ARC2.01: Users will be able to select options(OBH/NBH/Bakul/Parijaat) from the dropdown and view all listed machines in the selected hostel along with their current status.

Acceptance Tests for REQ3:
ATC3.01: Ensure each valid user will be able to view the list of all washing machines placed in their respective hostel along with their current status(vacant/occupied/out of order).
ATC3.02: Any washing machine which is ready to use and currently vacant must show status as vacant on the dashboard.
ATC3.03: Any washing machine which is currently running must show status as occupied on the dashboard.
ATC3.04: Any washing machine which is not in working condition must show status as out of order on the dashboard.

Acceptance Test for REQ4:
ATC4.01: Ensure the status of the washing machine should be occupied on the dashboard.
ATC4.02: Verify that the time remaining shown on the dashboard must be in accordance with the actual remaining time flashing on the washing machine display.

Acceptance Test for REQ5:
ATC5.01: When a user selects any date on the app to check booking availability, only time slots that are currently booked by any other users should be marked as blocked. All other time slots should be shown as free to users.

Acceptance Test for REQ6:
ATC6.01: Once users have booked the slot either using date and time or assisted booking feature if the booking is confirmed then the calendar should mark that slot as booked otherwise ask a user to try with another slot.
ATC6.02: App should not allow a user to book a slot that is already booked by another user.

Acceptance Test for REQ7:
ATC7.01: Once a reserved slot is canceled it should be shown as a free slot on the app. Any user now should be able to reserve this slot.

Acceptance Test for REQ8:
ATC8.01: User tries to book slots more that it is allowed(Fail)

Acceptance Test for REQ9:
ATC9.01: User tries to enter time duration more than the maximum allowed duration while booking the slot. (Fail)
ATC9.02: User tries to enter duration within maximum allowed duration while booking the slots. (Pass)

Acceptance Test for REQ10:
ATC10.01: If the remainder is set, the user should get notification about its reserved slot 15min before the start time and 5min before the end time

e. **What This Project is Not About:**
This project is not about setting up camera modules for the washeteria. The system is developed with the assumption that the appropriate camera is in place and the images are being written to appropriate locations in the backend server. The project starts by processing these already available images. However a suitable hardware setup could be done in order to extend the project's functionality and enable this feature.

4. Proposed Architecture

There are many ways to implement the system each with its own pros and cons. Here we are going to describe what and why of the chosen architecture.

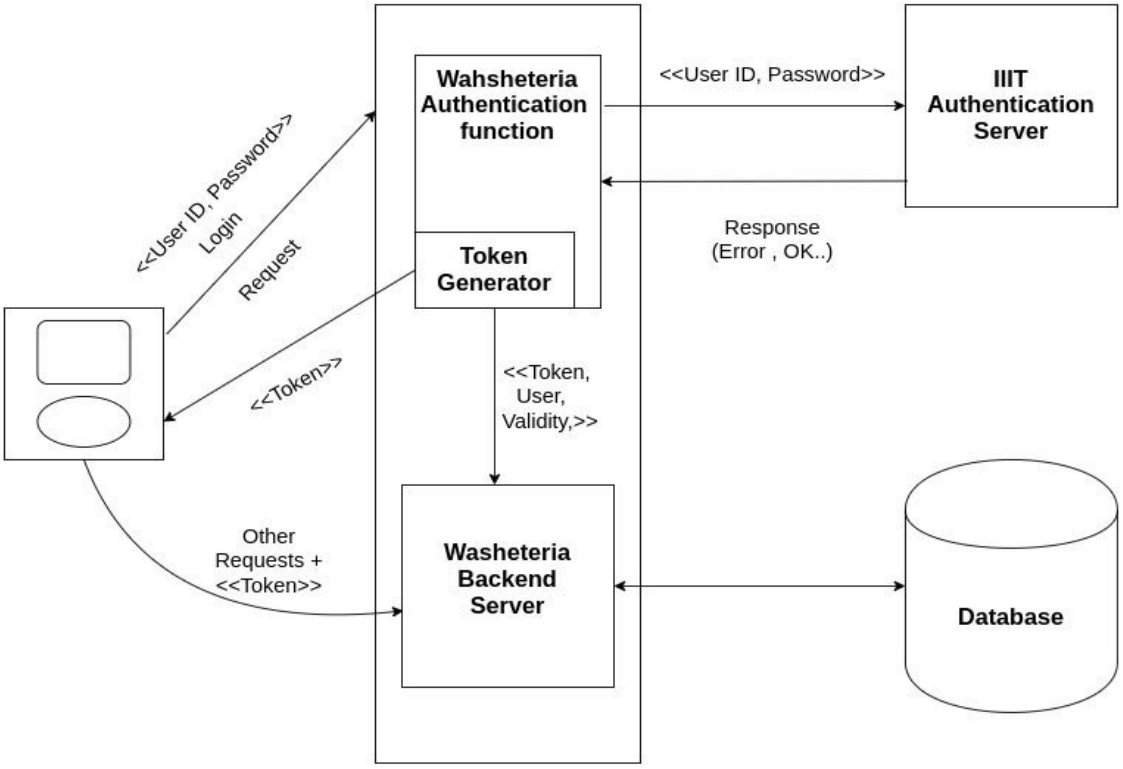
Actors

1. User: The client who will use this system.
2. Assistant: A component of the system which reserves slots on behalf of the user.
3. Server / Database: Stores all information and facilitates requests made by the user.

Use cases

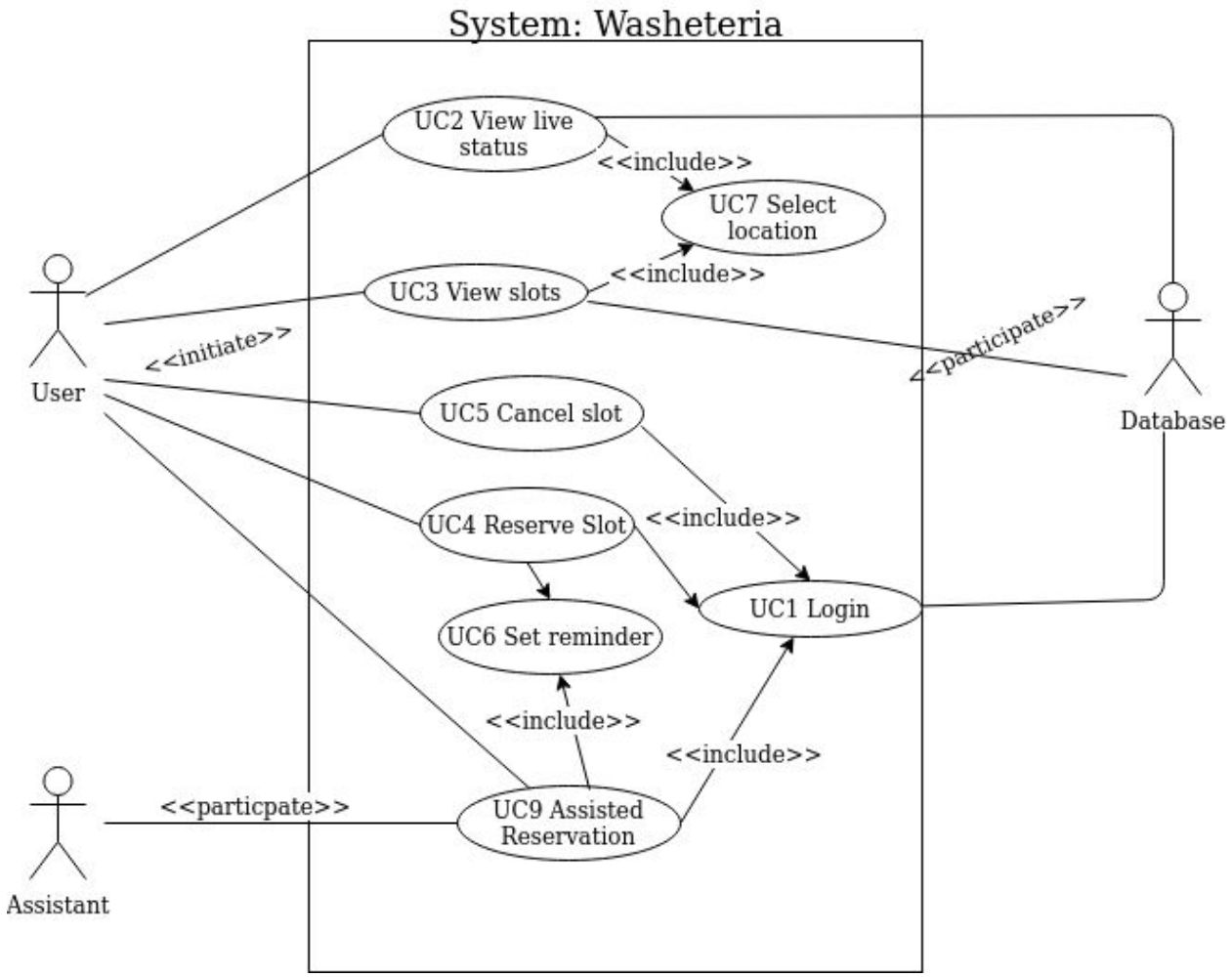
1. UC1: Login [Derived from REQ1]
2. UC2: View live status: [REQ3, REQ4]
3. UC3: View reservations and free slots[REQ5]
4. UC4: Reserve slot [REQ5, REQ6, REQ8, REQ9]
5. UC5: Cancel slot [REQ7]
6. UC6: Set reminder [REQ10]
7. UC7: Select location of washeteria [REQ2]
8. UC8: Confirm occupancy.[REQ6, REQ7]
9. UC9: Assisted reservation[REQ6]

Overall System Architecture Diagram:



Overall System Architecture

5. Use Case Diagram:



Fully-Dressed Description of Use Cases

From here onwards we consider the end-user to be a **student** of IIIT & the Washeteria software as a **system.**

Use Case UC-1	**Login**
Related Requirements:	REQ1
Initiating Actor:	Student, Database
Actor's Goal:	Login to the washeteria system
Participating Actors:	End user, System
Preconditions:	Student is a registered user (has IIIT email-id)
Success End Condition:	A Student is authenticated and logged in to the system
Failed End Condition:	Student failed to login or declined due to wrong credentials
Extension Points:	None
Flow of Events for Main Success Scenario:	**-> 1. Student** inputs the IIIT email ID and password **<- 2. System** verifies the credentials by interacting with IIIT authentication server **<- 3. System** prompts a login success message **<- 4. System** reverts back to the logged in dashboard page

Use Case UC-2	**View live status**
Related Requirements:	REQ3, REQ4, UC7
Initiating Actor:	Student, Database
Actor's Goal:	To view the current status of the washing machines

	(vacant/ occupied/ malfunctioned)
Participating Actors:	Student / Washeteria system
Preconditions:	Student must have selected the location from the list (OBH/NBH/Parijat/Bakul)
Success End Condition:	Student can view the live status of the washing machines.(Occupied/ Vacant/ Malfunctioned)
Failed End Condition:	None
Extension Points:	None
Flow of Events for Main Success Scenario:	**-> 1. Student** selects the **View live status** button **<- 2.** The **System** displays the current status(Vacant/Occupied/Out of order) of the machines at the location selected in **step 1** of the Use Case **UC7 Select location** of washeteria

Use Case UC-3	**View reservation & free slots**
Related Requirements:	REQ5
Initiating Actor:	Student, Database
Actor's Goal:	View the status of the slots (Reserved/Free)
Participating Actors:	Student
Preconditions:	Student must have selected the location (OBH/NBH/Parijat/Bakul)
Success End Condition:	Student can view the reserved and free slots
Failed End Condition:	None
Extension Points:	None
Flow of Events for Main Success Scenario:	**-> 1. Student** clicks the **View slots** button from the dropped down menu at the dashboard page **<- 2.** The **System** displays the calendar with details of booked & free slots.

Use Case UC-4	**Reserve slot**
Related Requirements:	REQ5, REQ6, REQ8, REQ9
Initiating Actor:	Student, Database, Assistant
Actor's Goal:	Reserve a slot for using a washing machine
Participating Actors:	Student
Preconditions:	Student is a registered user Student must have logged in to the system Student must have selected the location(OBH/NBH/Parijat/Bakul)
Success End Condition:	Student reserved the slot
Failed End Condition:	Slot unavailable due to high demand at the same time Student has reached the maximum limit of reservations allowed within a day
Extension Points:	**Assistant Reservation**
Flow of Events for Main Success Scenario:	**-> 1. Student** clicks the **Book** button from the menu at the logged in dashboard page **<- 2. Student** enters the **Date, start time** and **end time** **<- 3.** **3.a.** If daily limits restriction is not violated, **system** reserves the slot and updates the database **3.b.** If daily limits restriction is violated, **system** prompts user, the maximum limit allowed per day **<- 4.** System prompts a success message

Flow of Events for Extensions (Alternate Scenarios):	<- **5. System** reverts back to the logged in dashboard **-> 1. Student** selects the Assisted Reservation use case **UC 9** **-> 2. System** prompts to input the day and time(morning/noon/evening) **-> 3. Student** provides the day and time(morning/noon/evening) **<- 4. System** checks the availability of the slot at the required day and time as mentioned in **step 3** above **<- 5.** **5.a System** reserves a slot and updates the database if a slot is available on the required day and time. **5.b** System prompts regarding the unavailability of the slot meeting the requirements in **step 3** and **System** goes to **step 2** above. **<- 6.** System prompts a success message **<- 7. System** reverts back to the logged in dashboard

Use Case UC-5	**Cancel slot**
Related Requirements:	REQ7
Initiating Actor:	Student
Actor's Goal:	Cancel a reserved slot
Participating Actors:	Student, Database
Preconditions:	Student is a registered user

	Student must be logged in to the system Student must have selected the location(OBH/NBH/Parijat/Bakul) Student must have a reserved slot at the selected location in Select location step
Success End Condition:	Student cancelled the slot successfully
Failed End Condition:	Student does not have a reserved slot at the current location (OBH/NBH/Parijat/Bakul)
Extension Points:	None
Flow of Events for Main Success Scenario:	**-> 1. Student** selects the **my reservations** option at the logged in dashboard **-> 2. Student** clicks the **cancel slot** button **<- 3. System** cancels the slot and updates the **database** **<- 4. System** prompts a success message **<- 5. System** reverts back to the logged in dashboard

Use Case UC-6	**Set reminder**
Related Requirements:	REQ10
Initiating Actor:	Student
Actor's Goal:	Set up a reminder for a reserved slot
Participating Actors:	Student
Preconditions:	Student has a reserved slot
Success End Condition:	Reminder is set
Failed End Condition:	Student does not have a reserved slot hence reminder cannot be set

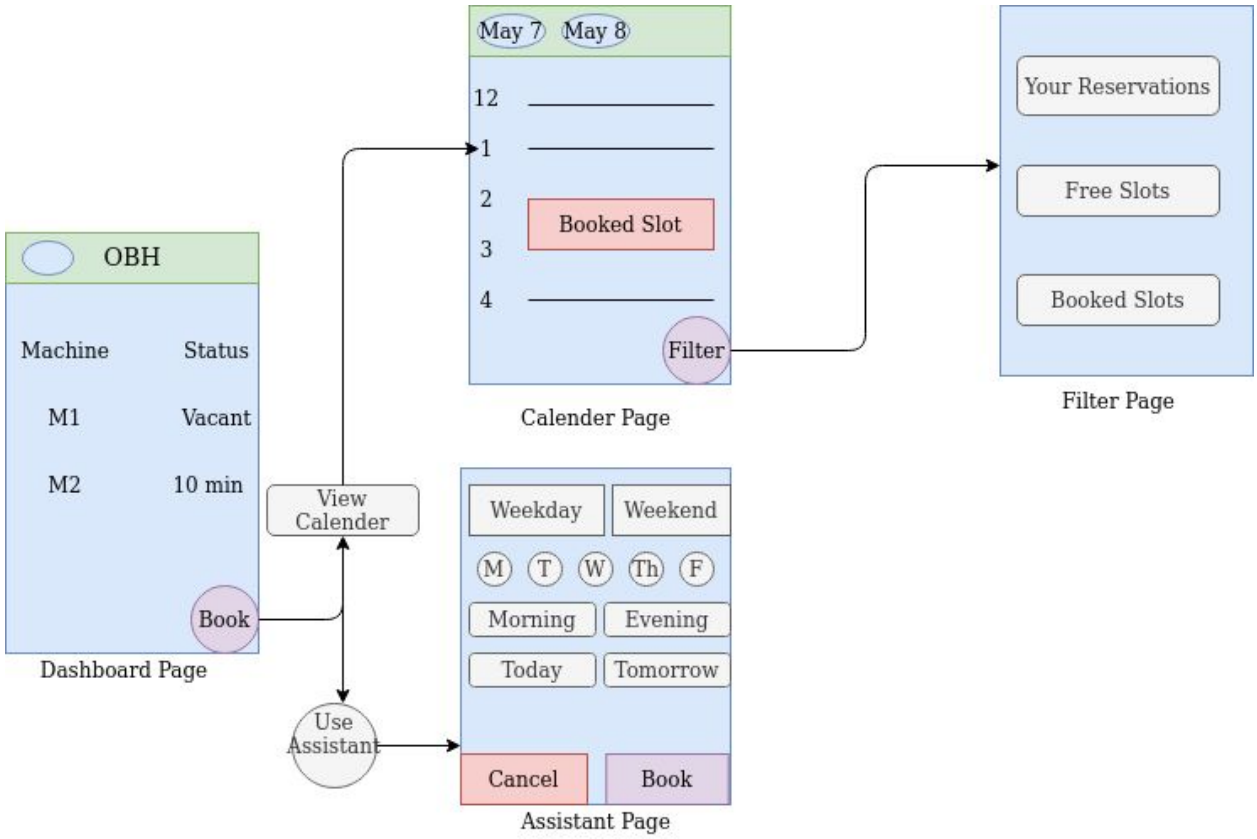
Extension Points:	
Flow of Events for Main Success Scenario:	**-> 1. Student** selects the set reminder option at the dashboard drop down menu **<- 2. System** displays the reserved slot of the user -> **3.** Student selects the slot and sets the reminder(15 minutes or custom time before the reserved slot kicks off) **<- 4.** System prompts a success message

Use Case UC-7	**Select Location of washeteria**
Related Requirements:	REQ2
Initiating Actor:	Student
Actor's Goal:	Select location from a list of locations where washing machines are installed (OBH/ NBH /Parijat/Bakul)
Participating Actors:	Student
Preconditions:	None
Success End Condition:	Student selects the location successfully
Failed End Condition:	None
Extension Points:	None
Flow of Events for Main Success Scenario:	-> **1. Student** selects the **Select Location** button from the dashboard menu <- **2. System** shows the options **View live status** Use case 2 and Use case 3 **View Slots**

Use Case UC-8	**Confirm using a slot**
Related Requirements:	REQ7, REQ3
Initiating Actor:	System
Actor's Goal:	Confirm the utilisation of a slot
Participating Actors:	Student, Database
Preconditions:	Student is a registered user Student must be logged in to the system Student must have a reserved slot
Success End Condition:	Student agrees or denies to use a slot
Failed End Condition:	Student failed to respond within a set limit
Extension Points:	None
Flow of Events for Main Success Scenario:	**<- 1. System** prompts the **student** thirty minutes(customizable) before the reserved slot whether he/she is actually utilising the reserved slot **-> 2. Student** replies with a **yes** or **no** **<- 3.** If the reply is no in above **step 2** the **system** updates the slot to free status in database

6. Detailed Design of Component 1(Frontend)

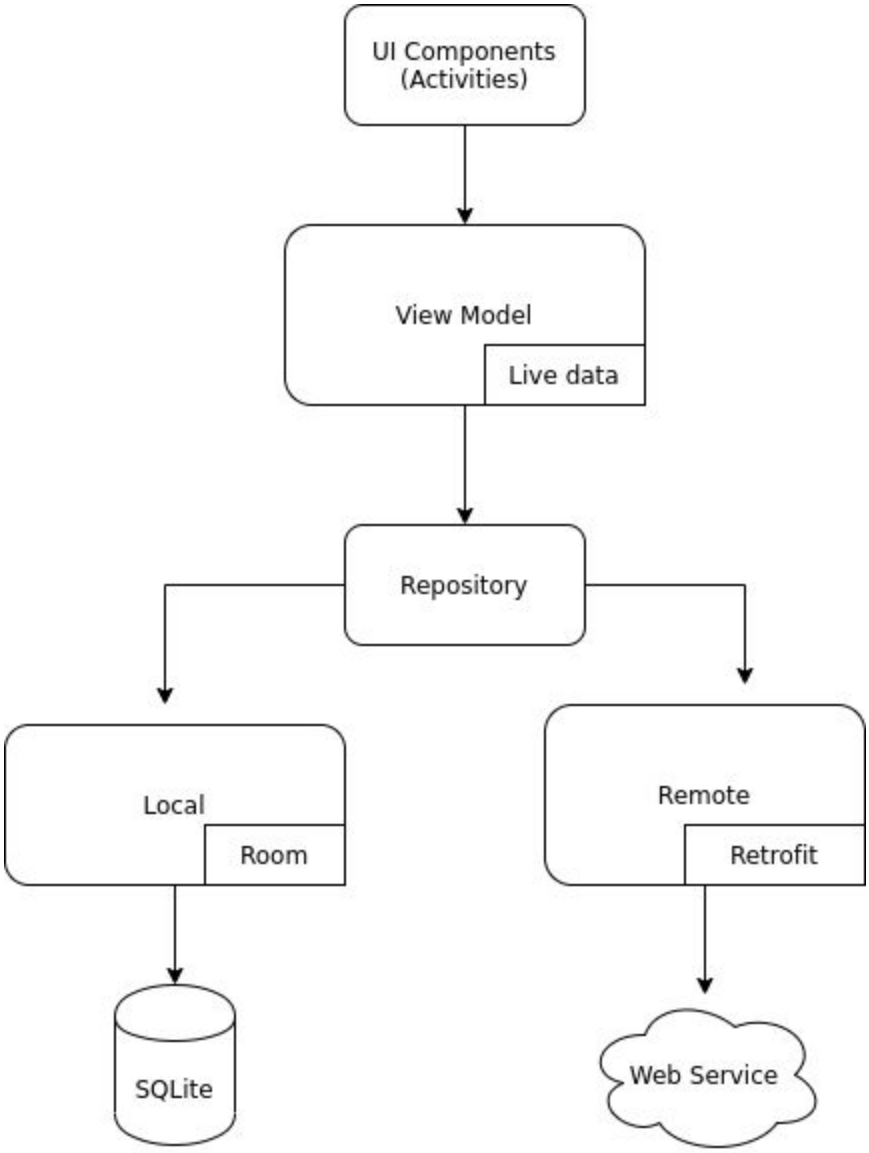
a. User interface specification



b. Architecture

The end user will be provided an Android app. The application will follow the android architecture component so as to follow the principle of *separation of concerns* and *driving UI from a model.*

The following image shows how all the modules will interact with one another.



This way the app will have a single source of truth and this is the internal database, the data has to be saved first or refreshed before they are displayed to the User Interface.

7. Detailed Design of Component 2 (Backend part)

I. Technologies Used

- Spring Boot Framework 2.2.6.RELEASE
- Hibernate 5 (ORM Provider)
- Java 11
- Apache Maven 3.6.3 (Dependency and Package Manager)
- Tomcat Server 9.0.x
- PostgreSQL (Database)

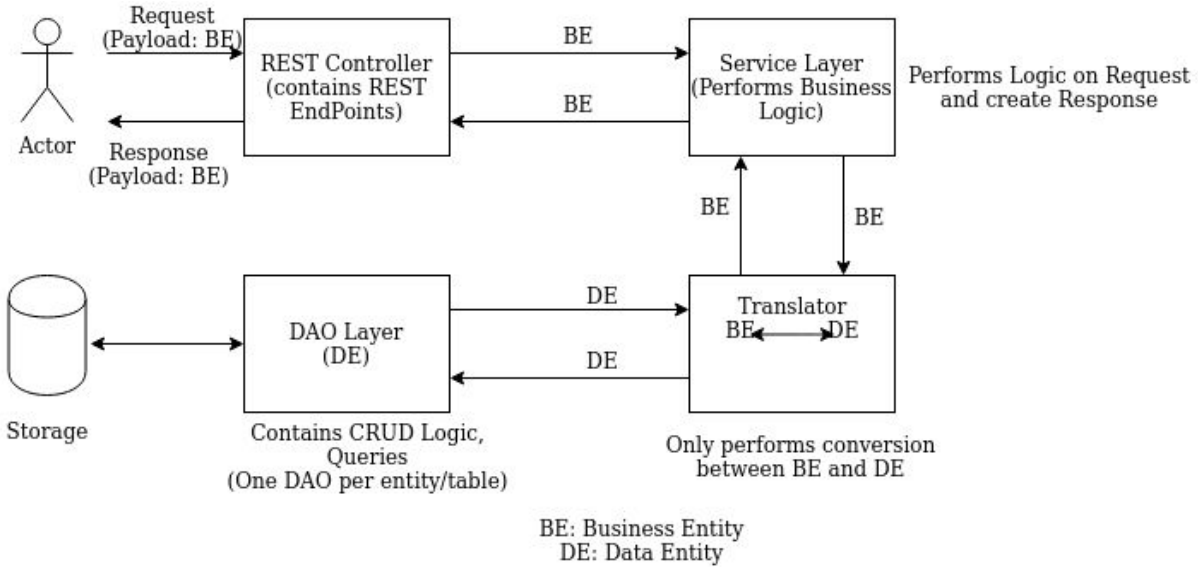
II. Package Structure and Naming Conventions

1. **Base package** : **com.iiith.washeteria**
    a. Contains Spring Servlet Init code

Rest of the packages should have a prefix : com.iiith.washeteria.

2. **com.iiith.washeteria.controller :** contains rest controllers
3. **com.iiith.washeteria.service :** interface for valid endpoint point methods
4. **com.iiith.washeteria.serviceImpl :** implementation of service methods and business logic
5. **com.iiith.washeteria.dao :** interface for persistance storage logic
6. **com.iiith.washeteria.daoImpl :** implementation of persistance logic
7. **com.iiith.washeteria.dataentities :** contains entities(POJOs) which will be stored in the DB. One to one mapping of tables and dataentities (hibernate entities)
8. **com.iiith.washeteria.businessentities :** POJOs which will be returned to the user in response, can contain extra attributes as per business needs
9. **com.iiith.washeteria.translater :** translates BE to DE and vice versa.

III. Architecture(Restful Services)



IV. Tentative RestFul Services Endpoints:

endpoint	Method	Request	Response	Response Failure

/getToken	POST	{“username”, “pass”}	{
			“token”: xbbvbfj
			“expiry”: datetime
			}
				401

/locations	GET		[{"id":"1", "name":"OBH"}, {"id":"2", "name":"NBH"}]	
/machines?location=	GET	Expected query param ‘location’	[{ “mid”: string “Name”: string, “status”:vacant/occupied/malfunctioned, remainingTime:time(remaining mins) },]	

/event	POST	{
		“mid”:string
		“locId”:string,
		
		“startTime”:datetime
		“endTime”:datetime
		}
			{
			“eventId”:
			“startTime”:datetime
			“endTime”:datetime
			“Mid”:string,
			“locationId”:location
			}
				409

/event?id=abc	DELETE	Expected query parameter	200OK	
/events?location=OBH&&stime=unixTime&&endTime=unixTime	GET	Query parameter	[{ “startTime”:datetime “endTime”:datetime “Mid”:string, “locationId”:location }]	

/events/assisted	POST	[
		{ startTime:
		endTime:
		unixtime
		},
		{
		startTime:unix,
		endTime
		}
		]
			{
			“eventId”:
			“startTime”:datetime
			“endTime”:datetime
			“Mid”:string,
			“locationId”:location
			}	

8. Appendix 1: Alternatives

The design choices along with the alternatives are listed below:

Should we make user login upfront before accessing any service or should login be required only when making a reservation: One way to design the system was to require the users to login before accessing any service, but this felt unnecessary. Instead of allowing users to view slots and live status of machines and only requiring them to sign in when they try to reserve a slot meant a better user experience. This decision significantly changes the system actors' interaction. This design decision is reflected in the use case diagram as only UC4: reserve slot and UC5: cancel slot require login.

Should clients directly interact with the IIIT auth server or via our Washeteria backend: Another design choice we had was whether to have the client send the login request to the IIIT auth server or to the Washeteria backend server which would verify the credentials sent by client by interacting with IIIT auth server. Both these choices have their own issues.

If we go with choice 1 then any change to the authentication mechanism would require rolling out an update to the clients and would also limit the places where this system could be deployed. Also, choice 1 has a flaw in that the client code be reverse engineered and then dummy authentication tokens could be passed to the Washeteria backend and backend would have no mechanism to distinguish valid requests from invalid requests.

If we go with choice 2 then washeteria backend code will be tightly coupled with the authentication system.
We decided to go with a modification of choice 2.

9. Appendix 2: Sequence diagram