Washeteria

SE Course Project

Team No 11

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

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.

Product Features

- User can access the application using a mobile phone.
- Current status : User will be able to view current status of each machine installed in respective hostels.
- View slots (Calendar):App provides a calendar like view where users can view reserved/free slots in any date.
- Reserve / Cancel a slot
- Notification before a reserved slot.
- Scheduling assistant (Automatically suggests free slots)

Innovations

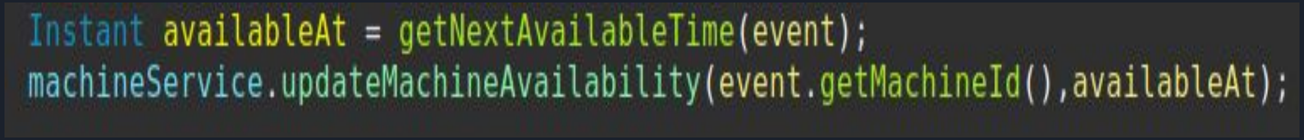
- **Ready to use Application :** We present a first cut, one stop solution for reserving machines for laundry at college.
- **Assisted Booking:** Leave it to us to find you a best slot for washing your clothes. Ofcourse we will ask for your preferences
- **Seamless UX design:** Washeteria provides you with a seamless UX to fulfill all you laundry needs.
- **Secure and Fault Tolerant** : Reverse proxy and token based authentication is used for authorization and backend servers are load balanced to support minimum application downtime.

Coding Principles

- Client Side uses MVVM architecture.
- App uses the local database as the single source of truth (driving UI from a model).
- Once app launches the user is immediately shown the locally cached data while the app refreshes in the background.
- App follows separation of concerns:
  - ❖ This means that every activity is only responsible for its UI elements where all its data is managed by the corresponding viewmodel.
  - ❖ Viewmodel is a kind of glue between View(UI) and Model(Data)
  - ❖ The actual data is managed by repository classes, so even the viewmodel doesn't know where the actual data comes from (local db or sharedprefs or backend APIs)

Coding Principles

- **Naming conventions:** All variables and functions follow camel case naming and are designated in a way that is meaningful to the context and present actors in the use case for which the code is written.



```
Instant availableAt = getNextAvailableTime(event);
machineService.updateMachineAvailability(event.getMachineId(),availableAt);
```

- **Clean code:** The code follows best practices of a clean code with single-responsibility functions and classes. Logic is broken down into smaller functions which are easy to comprehend. Comments wherever necessary.

Design Patterns:

- The **DAO design pattern** is used at layers of code which interact with the database.
- **DTO pattern** is used for data flow across the various layers of application and client interaction.
- **Facade pattern** is used to implement authentication.
- Complex object constructions use **builder pattern**(eg Assisted Event)
- The UI has got observers subscribed to local db changes.(**Observer pattern**)
- The client side uses **singleton pattern** for making network calls.
- **Connection Pooling:** All DB connections are handled via a connection pool and the application performs a JNDI lookup for obtaining connections.

Design Principles:

- **Distributed and Scalable:** The application is deployed on multiple load-balanced servers which can be distributed geographically.
- **Fault tolerant:** Database is deployed in master-slave with stream replication.
- **Interaction:** The backend exposes a set of restful apis for performing various actions (e.g. CRUD) on the available resources.
- **Layered-Architecture:** Various layers in the application perform their specific task such as client-interaction, business logic and database queries.
- **Security:** Application provides authentication via IIIT reverse proxy and authorization via generated token.

Project Management

- We separated our project code base into two separate repos:-
  - ❖ Main repo includes these two sub repos as git submodules.
  - ❖ This way the members involved in working on the frontend code base didn't need to worry about pulling the latest changes from backend repo.
- We used github project board for planning and tracking the tasks and to keep track of backlogs. Link to project board
- The project board is divided into 4 sections:- To Do, High Priority, Low Priority and Closed(Done)
- **Microsoft Teams** is used extensively to collaborate between team members.

Project Management

- **Build and Dependency Management:** Apache maven is used for managing dependency and creating deployable build artefact.
- **Code Management/ Version control:** Git and Github are used to collaborate code from multiple team members.
- **Google Jamboard** is used during brainstorming sessions while designing application architecture, setting REST APIs endpoints and finalizing application functionality.
- **Integration and Testing**: Used ngrok to create a secure tunnel for localhost so that server can be publicly accessed from any client. Tested the application by simulation several users.

Future Improvements

- Application can use computer vision to identify the remaining time from the washing machine control panel image to provide real time status.
- IIIT-H CAS authentication can be used for user validation.
- Once Students started using app for machine booking, data can be collected about machines usage and analysis can be done and details can be publish which might help in better maintenance of washing machine.

Thank You