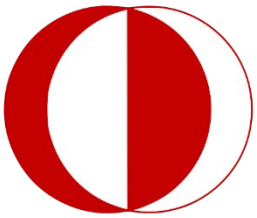

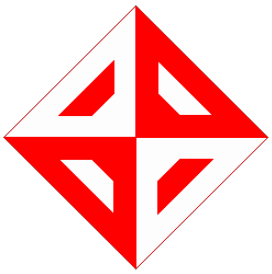

Software Design Overview Document

Group Members

Abdulkadir KILAVUZ

Aykut YARDIM

Cagdas FIL

Dogan SEZGIN

Advisor

Cevat SENER

Table of Contents

Figure Index

1 - Product Description

BikeSharing is a user-friendly rental bike project. The main purpose of the project is to increase bike usage in Turkey keeping up with the developing world standards in order to reduce traffic load and make people healthier. The developed project for this purpose, in briefly, exposes smart-lock system placed onto rentable bikes at the disposal of the user who tag the developed mobile application with the individual QR code on smart-lock.

By this way, the lock can be locked or unlocked at any time via the developed mobile app which keeps in contact since tagging. The whole process is subject to a payment system, starting with the user picking up the bike from the first particular area, and lasting until he ends the session in another area that is most convenient for user. These areas are all virtual zones that are not required the hardware, so they are easily relocatable. Providing a mobile application that allows the user to rent a bike, pay, open or close the lock at any time and to follow all these processes easily will create an alternative and easy-to-use transportation environment.

In these days when the safety, speed and using the time well are of great importance, the BikeSharing will increase people's interest in cycling.

2 - High Level System View

BikeSharing system consists of the Mobile Application, Cloud Server and BSDatabase. Both Mobile Application and BSDatabase are in contact with each other by communicating with Cloud Server. Mobile Application is the interface that provides to transmit data between user and the system. BSDatabase is the place where all the data are stored. Cloud Server is the bridge among the BSDatabase and Mobile Application.

As the external points the Smart-Lock is connected to cloud server to take information from Database which can be controlled by Mobile Application and it decides whether its situation is open or close. Payment services are performed via the Mobile Application and make thr Mobile Application collect user payment data in BSDatabase. Besides, the system management is the admin side of the project which provides configurations ability on the system. In addition, admins can display logs and results.

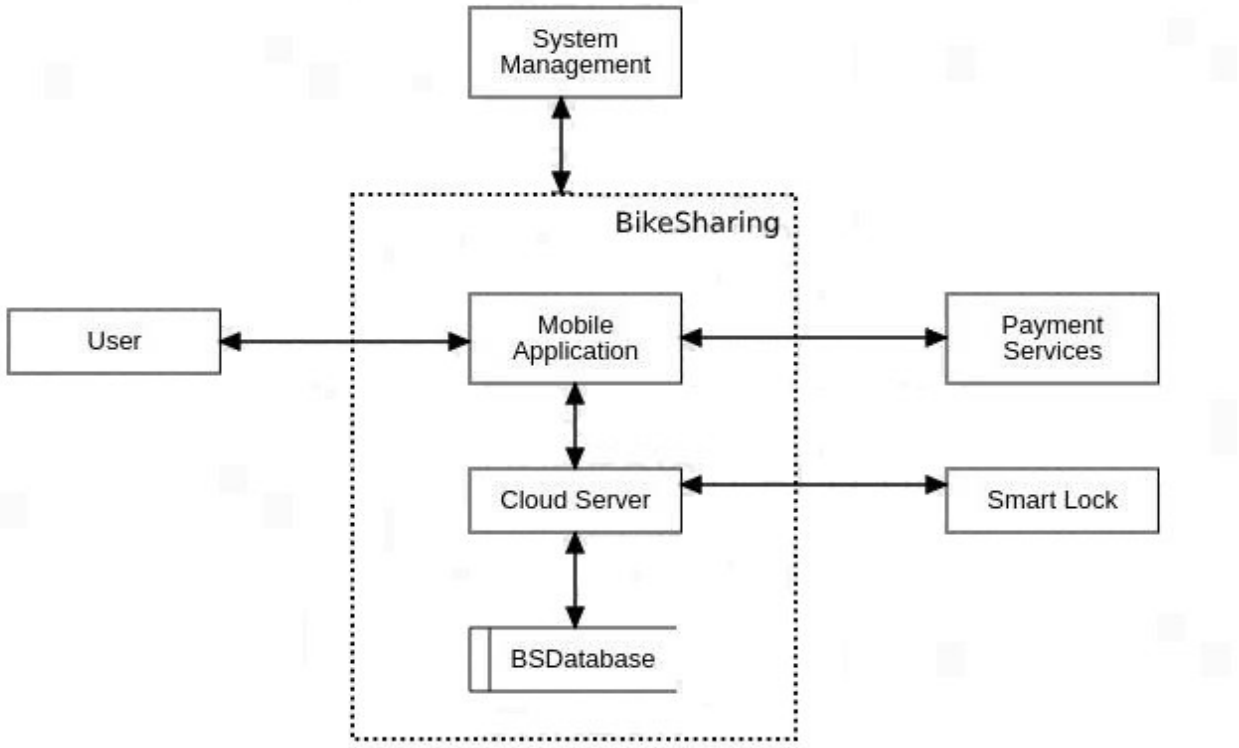

Figure 1: Context Diagram

3 - Overall Design

Overall design of the project is explained with two point of view in following subsections. In the first part, overall architecture of the project is presented with detailed description. Afterwards, detailed design of the project including two sequence diagrams is provided.

3.1 - High Level Design

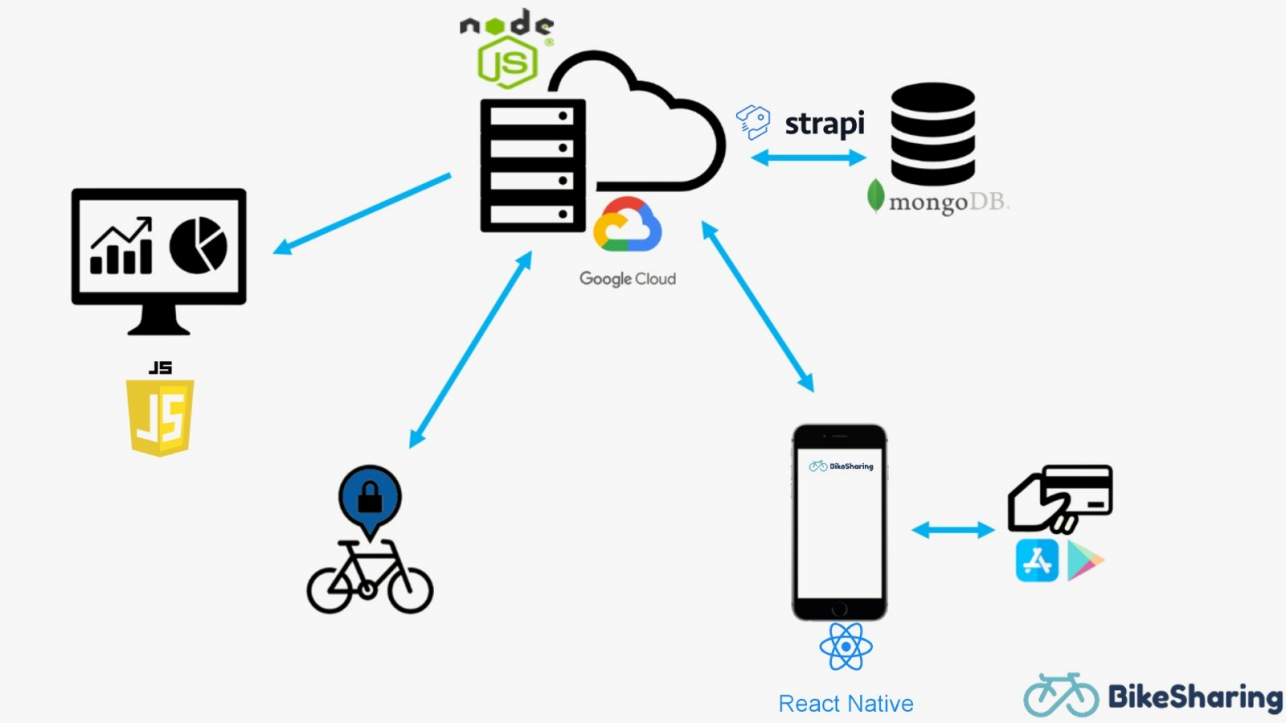

Figure 2: System Component and Their Interactions with Overall Architecture

The overall architecture consists of five internal components and one external component. There is a remote server which is centrally located in the system. The remote server is a cloud server that is based upon GCloud platform. The remote server has connection with a database which is also a cloud platform. For database, we preferred to use MongoDB which is a document-oriented NoSQL database used for high volume data storage.

In the overall architecture, there is a mobile application component which is the component that connects users to the system. In the mobile application, mobile application stores(App Store, Google Play) are used as an external service for payment operations. The payment process is integrated with the system via these mobile application stores.

The only hardware component in the overall architecture is the smart lock. The smart lock is in communication with the remote server. The mobile application communicates with smart lock through the remote server for basic operations(lock/unlock). Also, the remote server collects GPS tracking data from the smart lock concurrently.

The last component of the architecture is the Dashboard. Dashboard is an administrative component that displays detailed system information. The dashboard takes data from the database via remote server and shows derived statistics by processing and analyzing the data.

3.2 - Detailed Design

Beside adding or withdrawing money operations provided by BikeSharing with user-friendly interface, renting or returning operations of users are the main aspects of BikeSharing. Users in virtual zones controlled by GPS tracking can rent or return a bike using mobile application which calls ‘startSession’ or ‘endSession’ methods implemented on back-end side to start or end a session for the related user.

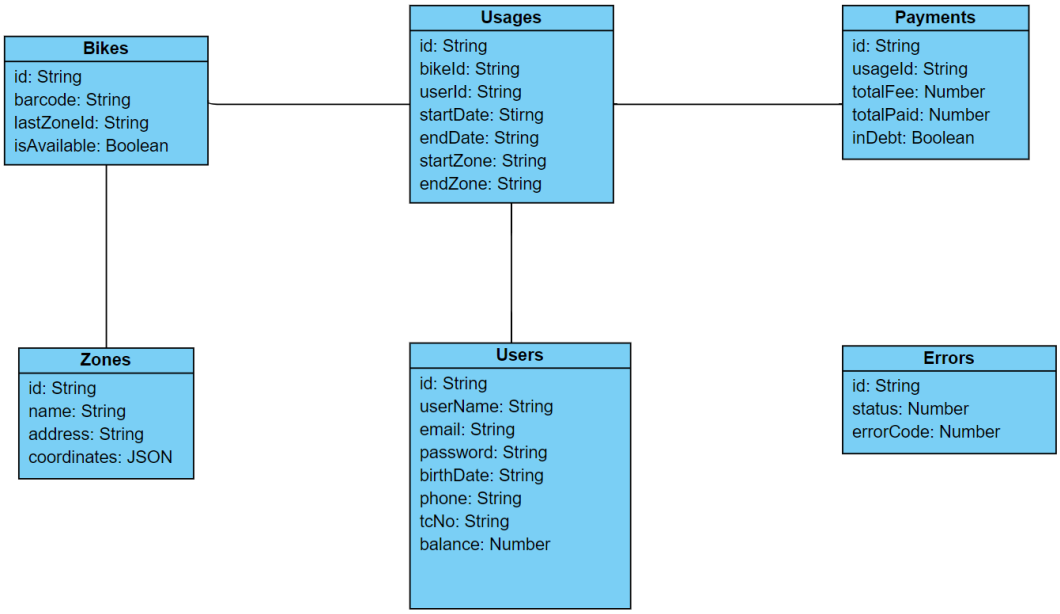

Figure 3: Class Diagram of BSDatabase

The class diagram in figure 3 illustrates the BSDatabase collection schema to examine the renting and the returning operations with ‘startSession’ and ‘endSession’ methods.

3.2.1 - Renting Operation

Whenever a user scans a QR code to rent a bike on home page, the ‘startSession’ method implemented on back-end side is called to start a session.

Workflow of Renting a Bike

- **Check QR code validation:** Querying to ‘bikes’ collection to find the related bike with scanned QR code.
- **Check bike's availability:** Querying to ‘bikes’ collection to check the related bike is available or not currently.
- **Check user's open sessions:** Querying to ‘usages’ collection to check the related user has open session or not currently.
- **Check user balance:** Querying to ‘users’ collection to check the related user has enough balance or not currently to rent a bike.
- **Create a usage record:** Adding a usage record to ‘usages’ collection with a unique id including start docker and start time information.
- **Update bike availability:** Updating the availability field of the related bike on ‘bikes’ collection.

The back-end code is implemented in a certain template that all cases have systematic responses for front-end side. All cases above has a unique error code if the response is not equal to ‘200’ and according to these error codes the related response message is shown end user.

Sequence Diagram of Renting Bike Operation

The sequence diagram in Figure 4 below, shows the actors, classes and their interactions with each other during a renting operation. Notice that, BSServer refers both the remote server which listens mobile application’s requests and responses them with back-end code implementation, and BSDatabase which stores the data controlled or manipulated by back-end services.

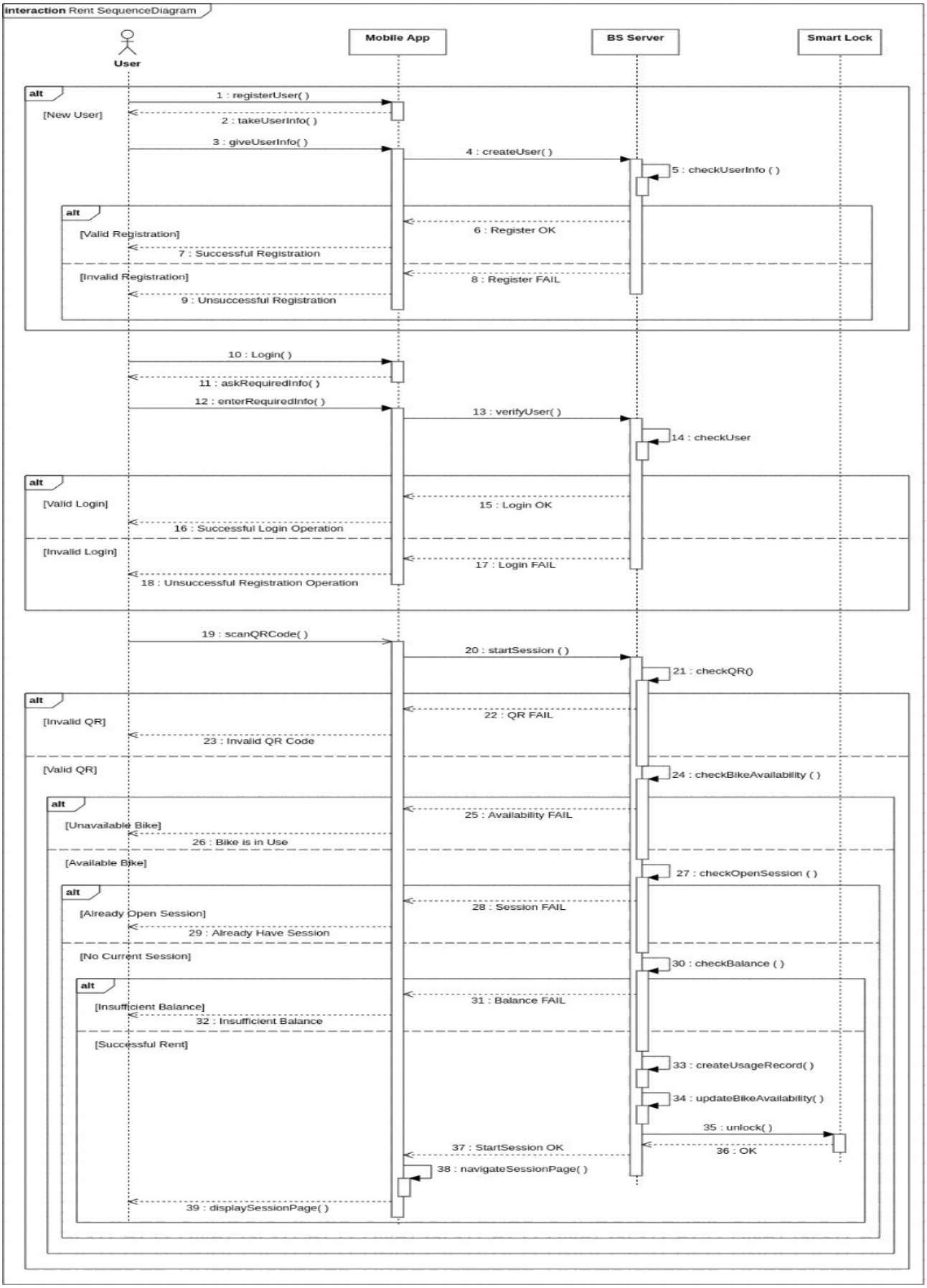

Figure 4: Sequence Diagram of Renting Operation

3.2.1 - Returning Operation

Whenever a user clicks end session button on session page, the ‘endSession’ method implemented on back-end side is called to end session.

Workflow of Returning a Bike

- **Check user has a bike currently:** Querying to ‘usages’ collection to check the related user has open session or not currently.
- **Update usage record:** Updating the end docker and end date fields of the related usage on ‘usages’ collection.
- **Find total payment:** Calculating total payment amount according to usage time.
- **Check user balance:** Checking user balance to specify having a debt condition.
- **Update user balance:** Updating user balance field of the related user on 'users’ collection.
- **Create a payment record:** Adding a payment record to ‘payments’ collection with a unique id including total fee and the amount paid currently.

The back-end code is implemented in a certain template that all cases have systematic responses for front-end side. All cases above has a unique error code if the response is not equal to ‘200’ and according to these error codes the related response message is shown end user.

Sequence Diagram of Renting Bike Operation

The sequence diagram in Figure 5 below, shows the actors, classes and their interactions with each other during a returning operation. Notice that, BSServer refers both the remote server which listens mobile application’s requests and responses them with back-end code implementation, and BSDatabase which stores the data controlled or manipulated by back-end services.

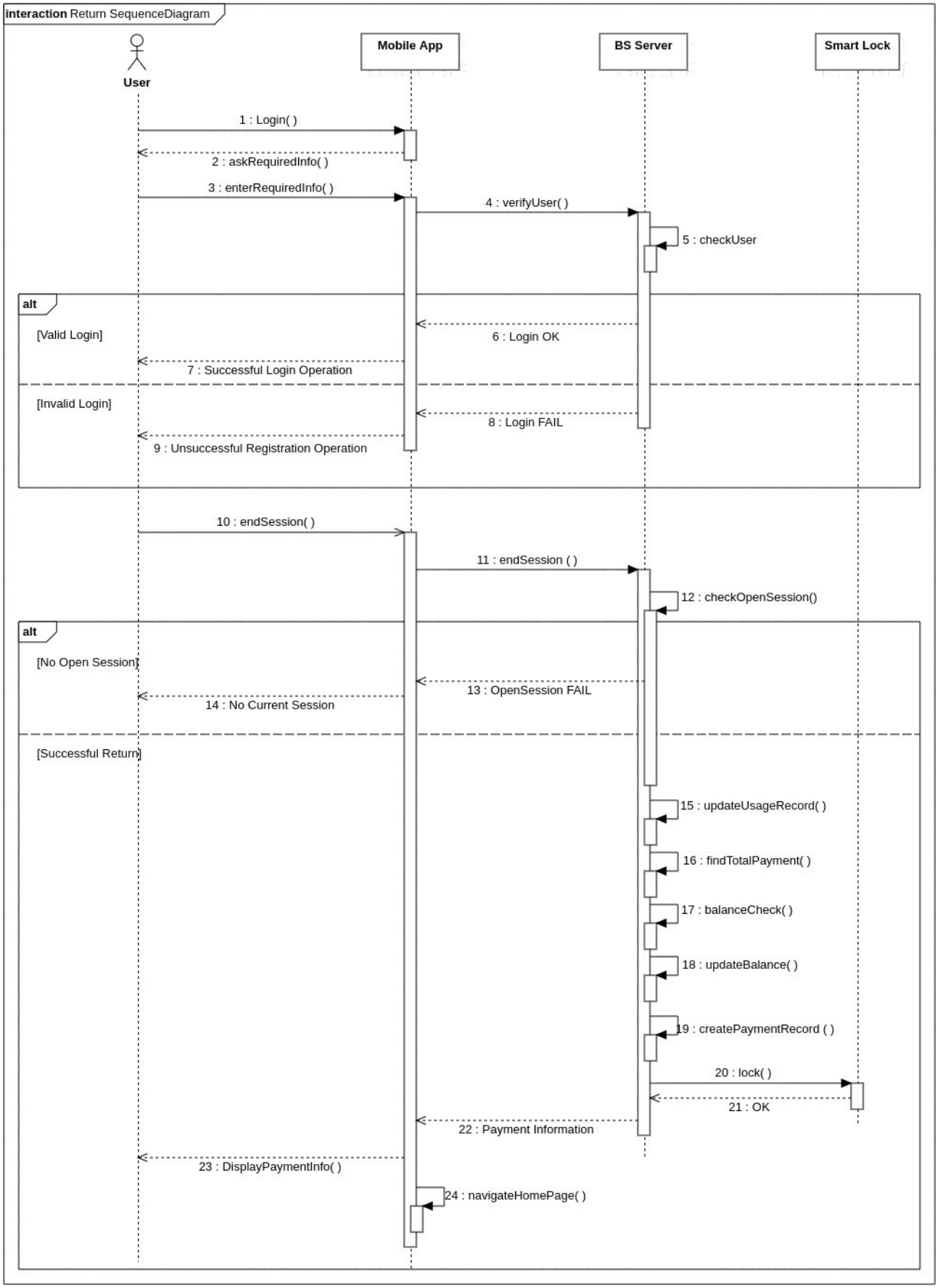

Figure 5: Sequence Diagram of Returning Operation

4 - Alternative Design Options

- We use Strapi instead of traditional usage of NodeJS because Strapi provides headless CMS which is a back-end only content management system built from the ground up as a content repository that makes content accessible via a RESTful API for display on any device.

- At the beginning of the project we thought that there would be physical dockers but it would be more expensive than virtual zone that we decided later as an alternative solution.

- For front-end, we use React Native as a cross platform that creates native apps for Android and IOS instead of implementing app for these platforms separately.

- Instead of creating a bare mobile application with React Native, we used Expo which is a management system for mobile applications that are built with React Native. With Expo tools, services, and React, we are able to build, deploy, and quickly iterate on native Android, iOS, and web apps from the same JavaScript codebase. There is no need to touch Xcode or Android Studio.

- We use Cloud server instead of remote server since it is more secure and has additional features that we may need to implement manually on virtual server.

5- Appendix

Abbreviation	Description
BS	BikeSharing System
User	General System User (Registered with e-mail address or phone number)
BSServer	The Remote Server of BikeSharing that Listens Mobile App. Requests with node.js Back-end implementation
BSDatabase	The Database Collection of BikeSharing that Stores Data for Back-end Queries on MongoDB platform

6- References

- UML Component Figure:

  https://www.uml-diagrams.org/component-diagrams.html

- UML Class Diagram:

  https://www.uml-diagrams.org/class-diagrams.html

- UML Sequence Diagram:

  https://www.uml-diagrams.org/sequence-diagrams.html