Architecture Design Document
Smart Locker 1.0

Daniel Chan
Ujjwal Khanal
Jonathan Laroco
Douglas Parker

Table of Contents

Introduction

The smart locker is a storage unit that allows users to provide and retrieve items in an automated and secure fashion, with software built to facilitate this item-transfer process. The idea is to provide an automated item management system that allows the developers to custom-tailor the software to their business needs, for example a safe haven for Craigslist transactions or a system for managers and appropriate authorities to transfer work related devices (such as smartphones, computers, etc...) between employees. In addition to business logic, the smart locker will automate item inventory, allowing users to check items in and out based on their administrative permissions. This includes both logging the different items and providing a full trace of where the item has been and who currently has it. This document will provide a complete overview of the software architecture, including the necessary components and connectors in order to provide a clear structure on how the smart locker software will be built.

Purpose

The purpose of this document is to provide a clear architecture design for the smart locker system. Included in the document are the components and connectors related to the system that will later be used for API models that will provide basic functions performed by the smart locker system. Models are used in order to show how various procedures and data will flow. This document will also describe the functionalities of the smart locker system including: retrieving an item, providing an item, managing the locker system as well as providing a report log of the different transactions that have gone in and out of the smart locker. In addition, the design of non-functional requirements such as scalability, availability and maintainability of the system are discussed. Overall, this architectural design document will provide a blueprint on how the smart locker system will be built.

Scope

The scope of the smart locker includes the ability to automate inventory management. The automation of inventory management can mean multiple things, including providing administrative access in order to facilitate items going in and out of the system. Additional functions include allowing users the ability to provide and retrieve items, with appropriate logging in the system. This logging can provide a report of items (or assigned lockers) to managers and provides a complete log of the transactions

between the items and users. In addition, the smart locker system will expose an API that allows developers easy access to smart locker functionality in order to customize and tailor the system to individual business needs.

Glossary of Terms

The table below shows a list of terms that will be used through the smart locker system architecture design document.

Term	**Definition**
Locker	A storage unit containing a set of items and a single control panel.
Item	A single “thing” which can be stored inside a locker. Can vary in size (from a few inches to a few feet) and dimensions (cubic to rectangular to spherical).
Control Panel	A user interface attached to a locker.
Providing an item	A user “provides” an item by placing it in the locker.
Retrieving an item	A user “retrieves” an item by removing it from the locker.
Transaction	The act of a user providing or retrieving an item from a locker.
Manager	An individual with respect to a locker which he/she has administrative rights to. An individual may be the manager of more than one locker.
Non-Manager	An individual with respect to a locker which he/she does *not* have administrative rights to.
User	An individual who uses the system. A user can be considered a Manager to some lockers and Non-Manager to other lockers.
Key	A digital representation of a physical key, which allows one user to provide limited access to their access privileges to another user. Such as one user giving another the right to retrieve a particular device.
Main Server	Server that hosts the database and API functions

	developed by the developers
Mobile Application	The mobile application system allowing users to generate and use keys for the locker

Overview

This document fully explains the architecture for the smart locker software system. The following sections describe all aspects of the architecture:

- **Architectural Representation:** Discusses the design decisions as well as the approach to implementation.
- **Architectural Goals/Constraints:** Shows the architectural constraints including the functional and nonfunctional requirements.
- **System Components/Connectors:** Discusses the components and connectors that will be used in architecture.
- **Use Case View:** Discusses the different use cases and will contain analysis/design models which describe them.
- **Logical View:** Shows the design's logic, including certain decisions based on user input data as well as the outcome from those decisions.
- **Implementation/Deployment View:** Discusses how the system will be implemented and deployed.
- **Data View:** In depth detail on how the data will flow throughout the system.

Architectural Representation

In order to accurately represent the architecture, we have chosen to model the smart locker after three specific architectural styles: peer-to-peer, client/server, and object-oriented. The system will not be based purely on one of these styles but take bits and pieces of each to construct a wholly new architecture built for this particular project. The following section will describe how the smart locker team plans on utilizing all three architectural styles.

Architecture Overview

As described in the previous section, the three different architectural approaches we plan on using includes the peer-to-peer style, client/server style as well as an object oriented style. These three styles will be described in the diagrams and models as well as demonstrate how they will integrate with each other. The architecture can be described as three separate entities: the management system, the clients/peers (which are the different separate systems including the actual locker as well as the receiving individual which will have a system on their smartphone that acts as a separate peer) and the objects. The objects are more so within each peers, as in within each system that application is installed on (in our case this would be the locker as well as the user client smartphone). The next section will go into a little more detail on how these three architectural styles are used in regards to the smart locker system.

Architecture Summary

The idea to incorporate three different architectural styles (peer-to-peer, client/server, and object oriented) originates from the idea to build three different software systems that all interact with each other to provide the primary functionality. The three different systems include the management system, which in our case is the server and database backend, the locker system (peer/client) as well as the individual smartphone system (peer/client). Because of the ability to add in an API interface as well as management systems there is a need to incorporate a client/server management system. The client/server architectural style will not be a traditional client/server architecture, meaning the client will not be taking input based on the server, more so the server will provide available functions that will be used to support the peer to peer architecture. The idea is that the peers (or clients in our case) will be able to do certain functions based off of the server (the server will hold all API calls as well as API development which will align with the functional/nonfunctional requirements). Additionally we will add in object oriented styles for each system, containing certain features for each separate system. The objects we create with the different systems will provide data to be sent back to the main server in which users will be able to generate reports that will be used for management. This gives us the customized approach that allows our APIs to become customizable and be used in many unique ways.

In conclusion, the architecture will be a mix of the peer-to-peer, client/server and object oriented architecture styles. The three different systems: Main database server, locker system and the user mobile cell phone will be utilized as three different objects

communicating with each other in a peer-to-peer setting. The functions provided will be coming from the server directly and the objects (three different systems) will have separate functions that send data to the main server for management purposes such as providing reports as well as inventory analysis. With the architecture described in this section the smart locker system will need to be implemented on three different hardware systems, this includes the mobile application, the main server as well as the locker hardware. This will be described in the section below

Implementation Strategy

In order to complete the full architecture, the smart locker system needs to be deployed and implemented on several different hardware systems. As described in the previous sections the smart locker system will need to be implemented on a main server, mobile apps, as well as on locker hardware which will control the access to the items in the locker. There will also be API functions created on the server which allow data to be inserted and obtained from the database. An API layer will provide database access to the client applications, both first-party and third-party. This API layer will allow developers to implement their own business logic on top of the existing architecture.

In addition to the main server, which hosts the database and API functions, there will need to be a mobile application that will be implemented. In our case we will be developing for both Android and iPhone apps, meaning there will be two different mobile platforms that we will need to implement on. These mobile applications will have an architecture related to the main database server in which they will have local database hosting right on the mobile device. Data will be saved onto the local database and functions will be created in order to use the API's that are hosted on the main server. Functionality on the mobile platforms will primarily act as an interactive layer between the user and the main server.

The last implementation we will need to provide for the smart locker system is the actual locker system. Unlike the other systems, the smart locker will be the easiest to implement as it simply gets and receives data based on the mobile and main server systems. The main purpose of the smart locker system would be to lock and unlock the locker based on data provided by the main server and mobile systems.

In conclusion the system will need to be implemented on the main server, the Android and iOS platforms and the locker system. The main server consists of a main database and API functions that will be utilized by the mobile and locker clients. Mobile systems will have a local database and can be installed from the iOS and Android marketplaces respectively. Lastly, the locker system will be implemented on a stand

alone hardware platform. Software on the locker will communicate with the main server and open/close the lockers as appropriate.

Architectural Goals and Constraints

This section describes the smart locker's architectural goals and constraints which include some of the functional and nonfunctional requirements that make a significant impact on the smart locker system.

Technical Platform

As described in the previous section the three platforms that the smart locker system will be deployed on include: The main database server (to store data), mobile (to view data and interact with the locker) and the locker itself. The mobile applications can be downloaded on either the Google Play store (Android) or the Apple App store (iOS). Lastly the locker will have a hardware setup with a wifi enabled computer chip controlling the different lockers as well as a QR reader to read from the mobile application.

Basic User Functional Requirements

The user functional requirements includes normal users interacting with the different lockers. The functional requirements for the users include the following:

- **Create account:** A user can create an account through a specified portal.
- **Log in:** A user can log in to the system through a specified portal.
- **Retrieving an Item:** A user can retrieve items from a locker if they have the correct permission over that item.
- **Providing an Item:** A user can provide an item to the locker if they have permission to do so.
- **Requesting an Item:** A user can request an item from another user, even if they do not *necessarily* know which user currently has the item.
- **Giving an Item:** A user can give access to an item to another user, who can then pick it up from the locker at their convenience. The user receiving the item have or may not have requested it previously.

- **Code Generation:** A user will have the ability to generate a unique code or key that will be used for the locker.
- **Code Scanning:** A user will have the ability to scan a code, processing the key and unlocking the locker.

Manager Functional Requirements

Manager functional requirements allow administrators to pre-program their lockers' behavior as well as view transactional logs from the database. This provides a high level view to see current inventory in the locker, which users have which items, and the movement of the inventory over time. The following manager functional requirements include:

- **Log in:** A user can log in to the system through the administrator portal.
- **Give user access:** A manager can provide a user with access to particular item in a locker, or to provide their own item to the locker.
- **View Locker Inventory:** A manager can view all the items currently in a particular locker he/she has rights over.
- **View Item Transaction:** A manager can view all the transactions which have taken place in a particular locker he/she has rights over.
- **Locker Behavior Set Up:** A manager should be able to define the behavior for a locker he/she has rights over (should it behave like a library, a dropbox, a transfer, etc.).

Non-Functional Requirements

The non-functional requirements for the smart locker system allow the architecture to be able to scale out and fill in any gaps within the requirements that are needed. This includes basic functionalities including the need to maintain the system, scale the system out depending on the need of the business and applying security measures to make sure that the smart locker system is secure within its own network. The non functional requirements for the smart locker system are listed as follows:

- **Security:** The inventory system would be secured with transmission-level features as well as application-level features. For secure transmissions over the Internet, confidentiality, authenticity, and integrity are critical. Confidentiality means that only the desired parties are able to access and read a message.

Authenticity means that any message or action can be attributed to a particular user. Integrity means that any message cannot be altered after being sent. Encryption provides for all three of these requirements and enables secure transmission of information across the Internet. To protect the application itself from misuse, authentication and authorization will be used. Authentication requires that users must access the application using a secure username and a password. Authorization restricts users based on their role to only have access to specific actions for specific items. This functionality ensures that users are not able to do anything malicious within the bounds of the system.

- **Reliability:** Reliability is necessary for this application to ensure that items are not trapped within a locker, unable to be accessed because of a software failure behind the scenes. A hardware fallback should be provided in the form of a physical key given to the manager of a locker so he/she can manually open it if necessary without any backend support.
- **Scalability:** Inventory of the items can grow to very large amount with increased users who are taking direct interaction with the application. Inventory system should be scalable as per need with the devices it can hold as well as the number of growing active users. In addition, the system should be able to handle unlimited locker space meaning there will be no major performance issue is there are 2 locker spaces or 1,000,000 locker spaces.
- **Accessibility:** Individuals with physical impairments, such as blindness, deafness, or mobility restrictions should still be able to use the system. Talkback features in the mobile applications should allow impaired individuals to use them, while the locker will need to have a similar system implemented to allow impaired users to provide and retrieve items.
- **Usability:** All 1st part client platforms should be intuitive and easy to use, even for inexperienced users or those unfamiliar with the smart locker system.
- **Traceability:** The smart locker system should maintain logs of all transactions made by users as well as any errors which occur. Logs are made available to managers to track usage of various items. More in-depth logs are made available to the development team to debug failures in the system and ensure that it is working correctly.
- **Maintainability:** As additional requirements are added to future iterations of the smart locker system, development should be able to scale and handle these changes. Development environments and tools will be kept up to date to ensure that they are enabled for future work.
- **Public API Design:** The inventory system will have multiple different needs for different business logic. This includes providing a public API for developers in order for them to tailor the smart locker system to their specific needs. 1st party

clients can provide general functionality for most common use cases, but the public API enables 3rd party developers to implement their own domain-specific functionality on top of the existing architecture.

- **Documentation:** In order to provide a usable public API, it must be appropriately documented with all the necessary specifications. This should provide 3rd party developers with a smooth and simple path to integrate their own applications into the smart locker system.

Components,Connectors and Configuration

The following section describes the smart locker system architecture's components, connectors and configurations.

Components

This system is architected with a set of components, each of which provides a specific dedicated function.

- **Database:** A data storage system which contains all the data relevant to the system.
- **Main Server:** Provides a set of basic operations which can be performed and exposes them through a publicly available API. This would include actions such as "$USER retrieves $ITEM" or "$USER provides $ITEM" and queries such as "What $ITEMS does $USER have access to?" and "Is $USER a manager of $LOCKER?"
- **1st Party Clients:** Set of all client devices owned by the smart locker team. This includes the locker, Android app, iOS app, and web application. Each client fulfills a different purpose to the user and represents a "view" of the data, providing a way for the user to interact with it based on the specific device's use case. Additional clients may be added in the future, such as a mobile web interface, or any other yet-unknown platform.
- **Locker:** The locker allows users to provide and retrieve physical items. It would have an interface for users to interact with built into it and include a scanner to read QR codes generated by the mobile apps.
- **Mobile Apps:** The Android and iOS mobile apps allow users to manage items they have access to. This is where users can request and give items to each other. This is also where users can generate a QR code to claim an item from the locker.

- **Web App:** The web application is used by managers to perform their higher-level functionality. This is where a manager can set up a new locker and define its functionality as well as track the current status of items and lockers in the system. Managers can also view previous transactions which have been made within the system.
- **3rd Party Clients:** Set of all client systems *not* owned by the smart locker team. This could include any device set up by a third-party to take advantage of the smart locker system. This could include additional mobile apps which implement domain-specific functionality, or may instead include a set of third-party servers implementing the business logic for a completely separate application of which the smart locker system only plays a small part.

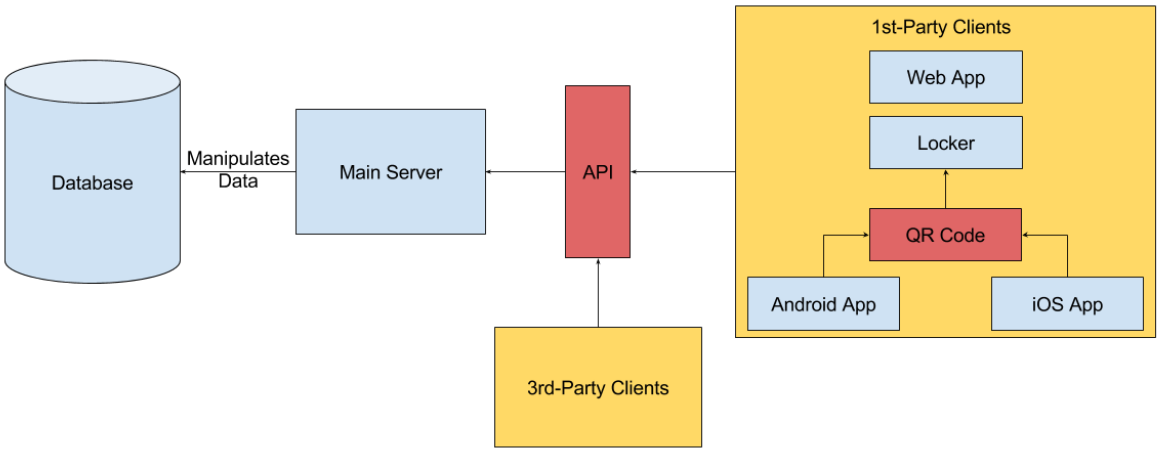
Connectors

Individual components are useless on their own, so connectors are necessary to combine their specific functionalities into a single coherent system.

- **API**: A specification of input and output formats to invoke specific operations on the internal data and parse the result. This might include “What items does $USER have access to?”, “What lockers does $MANAGER own?”, or “Transfer ownership of $ITEM from $USER_1 to $USER_2”.
- **QR Code:** Mobile apps are able to generate QR codes which can be scanned by the locker in order to authenticate users to retrieve items. This provides a mechanism for the mobile apps to communicate directly with the locker in a peer-to-peer manner.

Configuration

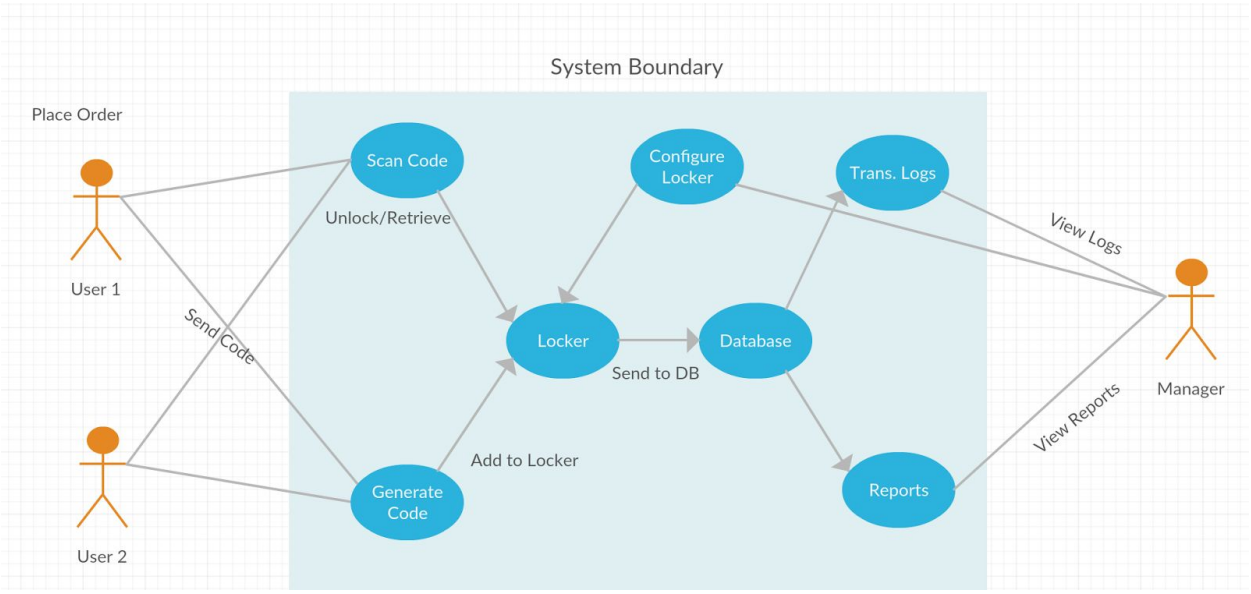
The following diagram shows the configuration that correlates with the system’s architecture. In addition it shows how the different components and connectors are related to each other.



Use Case View

The use case view consists of three different actors, user1, user2 and the manager user. User1 and user2 are users interacting with the smart locker system, allowing them to provide and retrieve items from the locker. User1 has the ability to scan a QR code, while user2 has the ability to generate and send a QR code to user1. Additionally, user2 can physically provide the item to the locker while user1 can unlock and physically retrieve the item from the locker. The transactions are made available to the manager actor. This actor is allowed to view logs and reports about the status of lockers which he/she has access to.

- **Item Management Report(s):** Reports will be provided that allow the manager to run against any locker and/or item which he/she has rights to. These reports provide exact details such as transactions logs as well as inventory movement in regards to items. The reports allow for inventory management as well as locker management. The reports will run on a date range, allowing users to see inventory and items based on a specific user defined date.

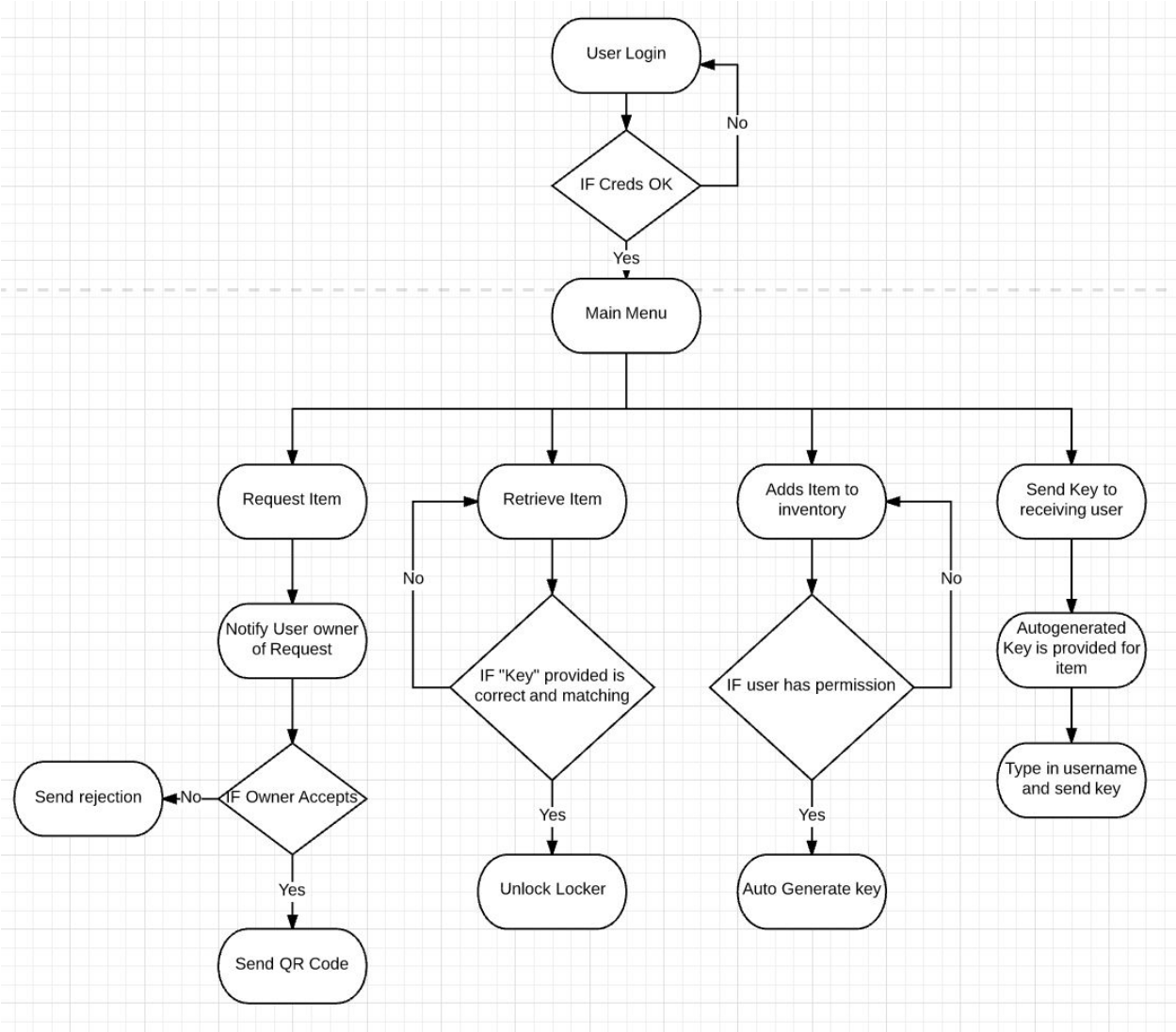


Process View

The process view shows how various functions relate to each other as well as a high-level view of the process. The two different activity diagrams, the user diagram and the manager diagram, are fairly simple illustrations which show the different user flows which can occur based on certain actions.

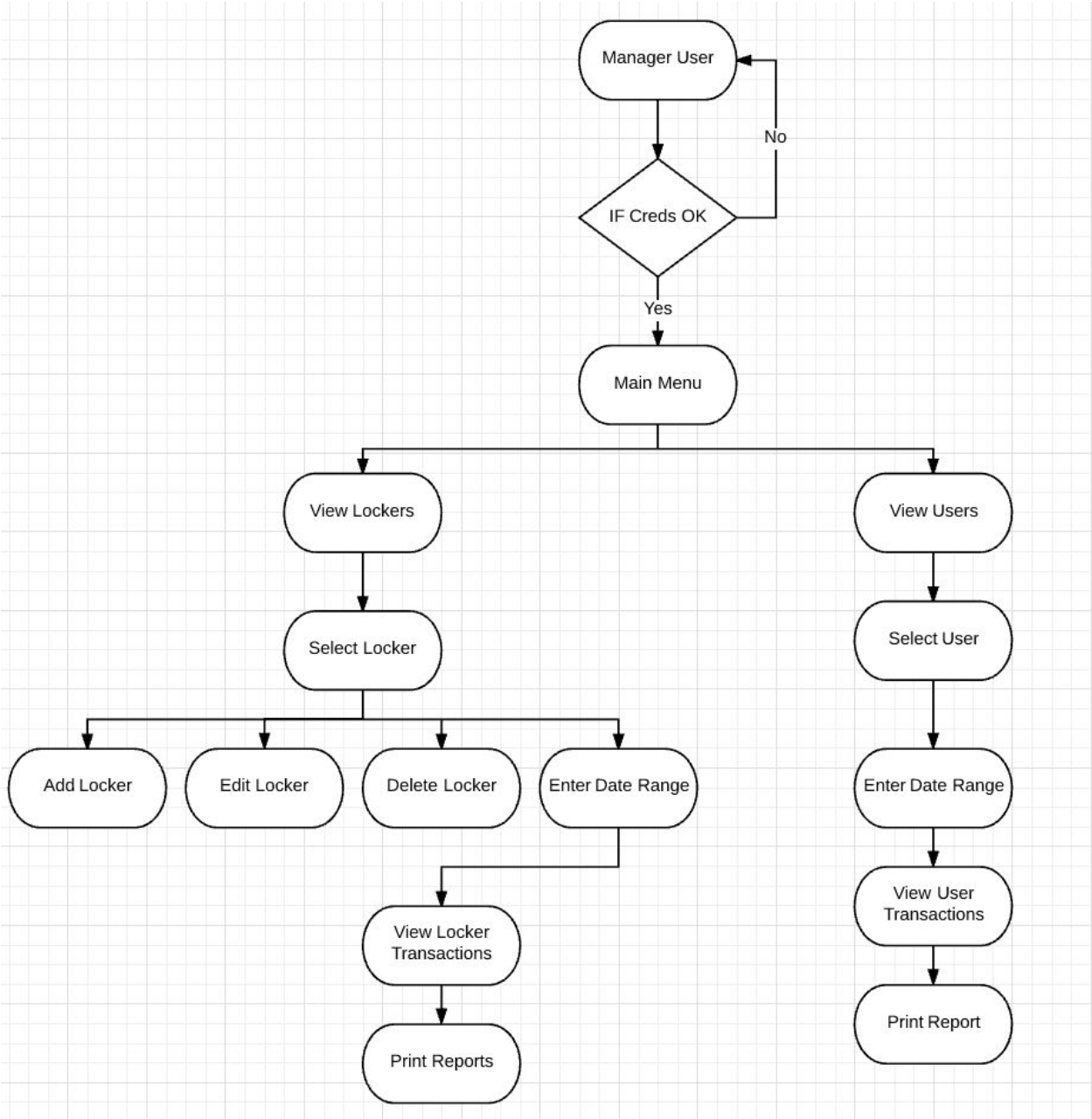
Basic User Activity Diagram

The basic user activity diagram demonstrates the four functional requirements and details how each action works in regards to the system. The user can request an item from another user, retrieve an item he/she has access to from its containing locker, provide an item to a locker, and send a key to an item to another user. There are permission checks for providing an item as well as retrieving an item. Other functions include generating a QR code and sending it to other individuals.



Manager User Activity Diagram

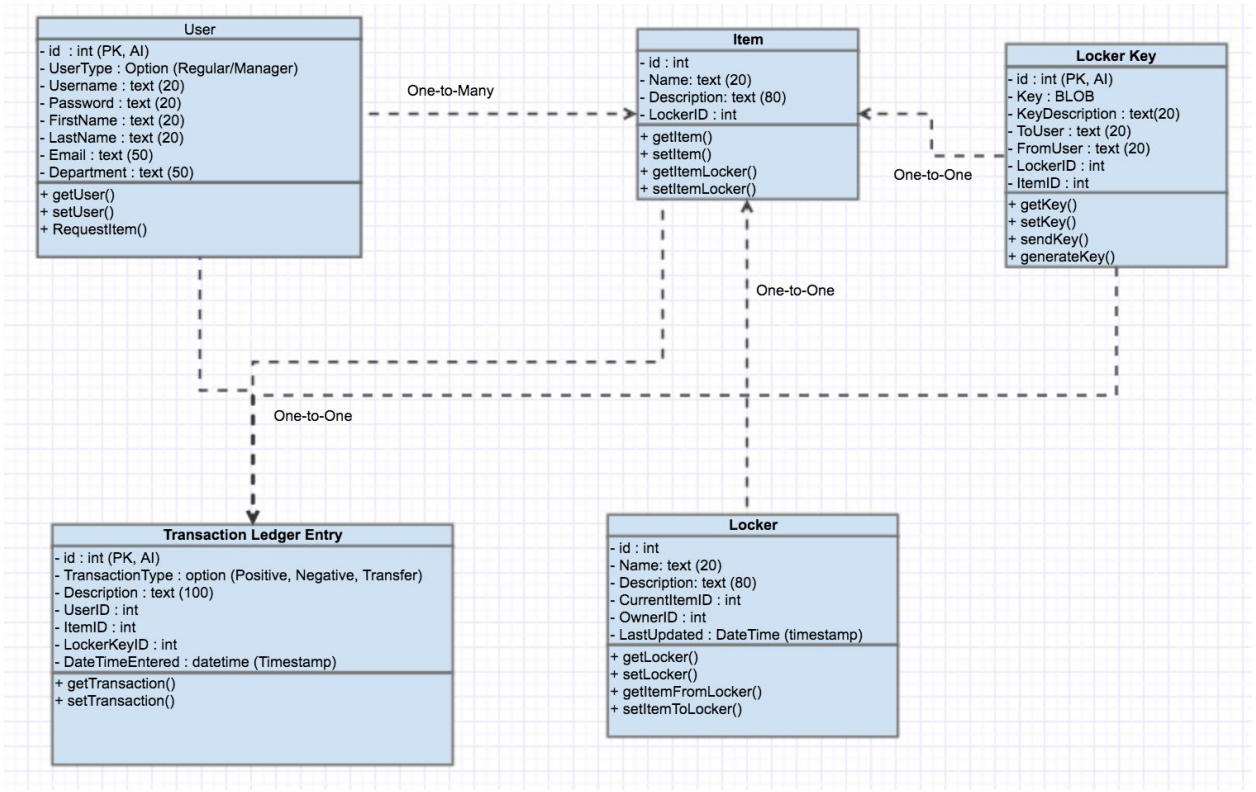
The manager user activity diagram provides views such as transaction logs as well as various reports. In addition, it allows the manager to configure lockers using functions such as add, edit and delete lockers.



Data Model View

The data model view has multiple views and tables including the user, QR Relation, Item and locker tables. Each of these tables represents specific data for the different entities including the ID, which is the universal primary key, as well as a name and description. The three master tables include the item, locker and user tables. The QR relation table holds the actual QR code and relative information, that is used to

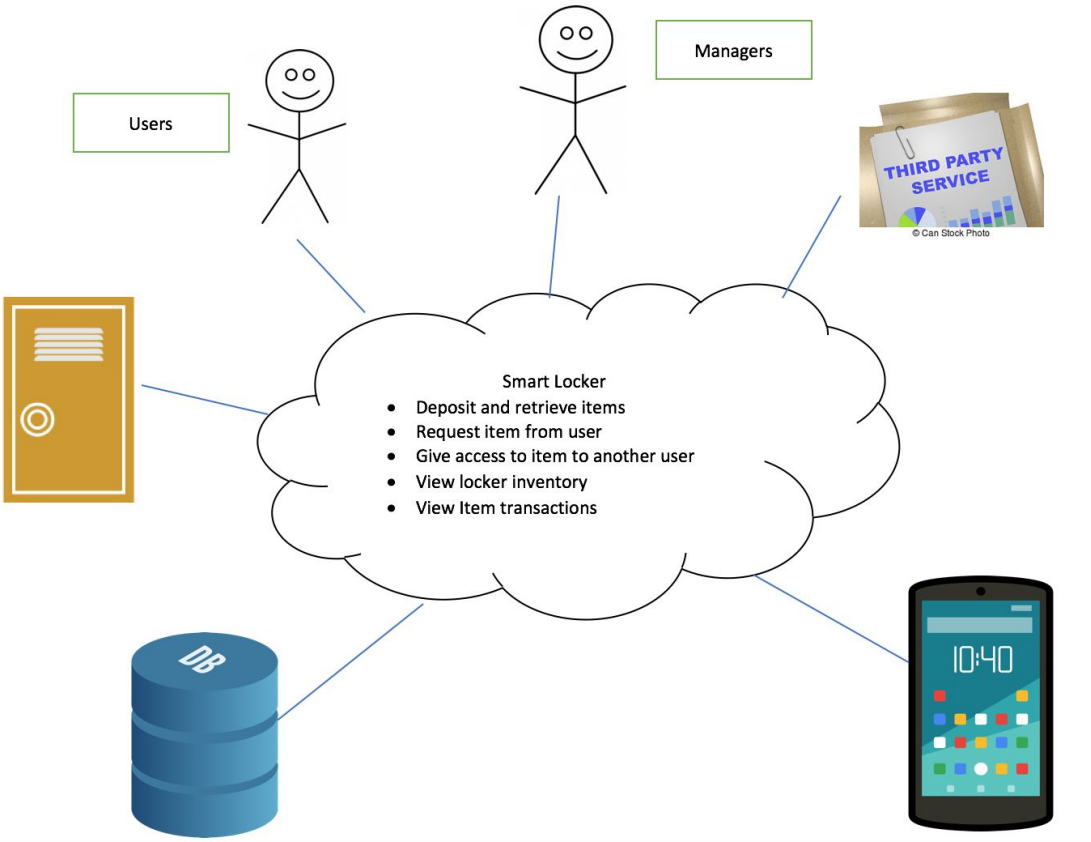
connect the item back to the user and vice versa. Inside of the QR code relation table are the item and user values connecting each QR code to an item and user in a one to make relationship. Lastly the transaction ledger entry is a transaction table that records each transaction between the item and the user. The transaction table includes a unique ID which is also auto incrementing and the primary key within the table. The transaction type includes how the item was moving and in what regards. This includes the specific transaction such as transfer, positive and negative adjustments into the locker. More importantly the date and time field includes when the transaction was logged onto the table. All of these tables make up the Smart Locker system and can be used for inventory management and reporting.



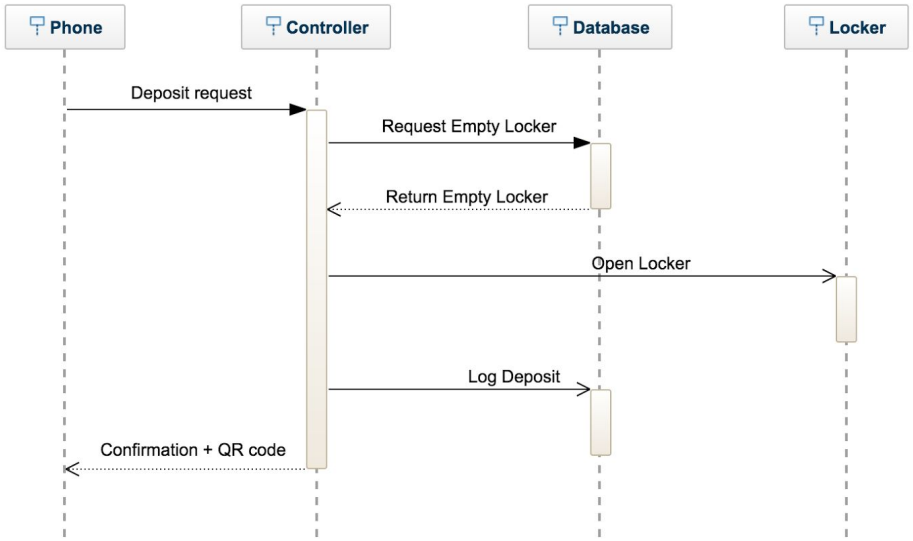
Models

The following section shows the different models representing the system's architecture, these models include the OV-1, and sequence diagram models.

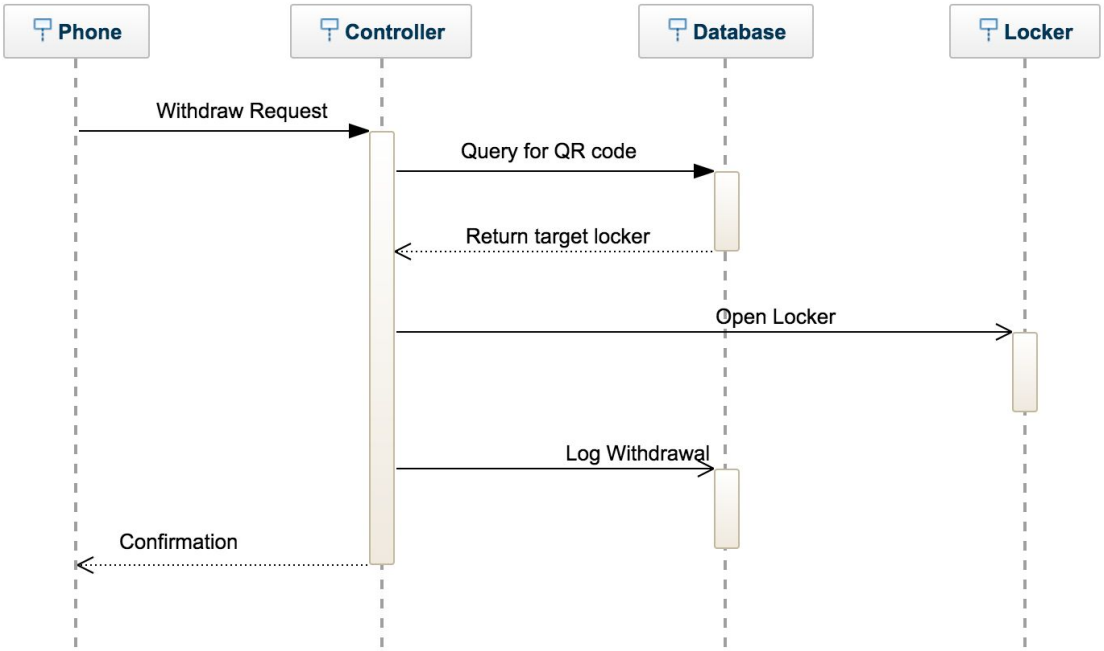
OV-1


Users
Managers
THIRD PARTY SERVICE
© Can Stock Photo
Smart Locker
Deposit and retrieve items
Request item from user
Give access to item to another user
View locker inventory
View Item transactions
DB
10:40

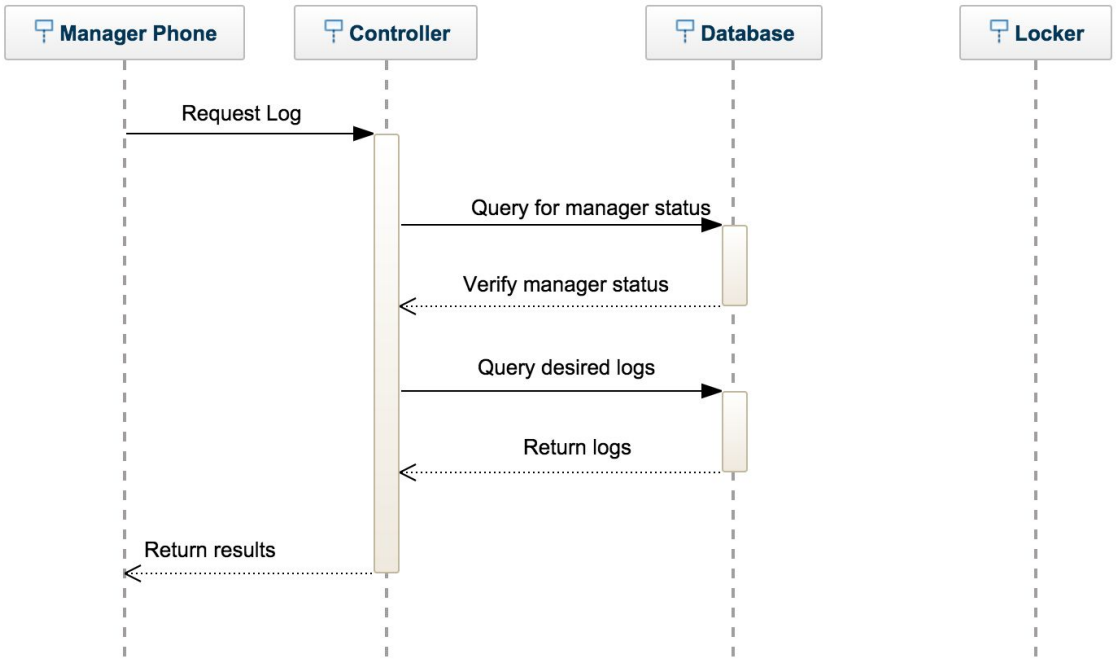
Sequence Diagram - Deposit



Sequence Diagram - Withdrawal



Sequence Diagram - Manager Request logs



Conclusion

In conclusion, the smart locker system is an automated item inventory system that allows the automated entry and exit of a physical item from a locker. The handling of an item is all automated, this includes specific item tracking and allows administrators to view who has an item and where an item has been in a given day. The purpose of the project is to provide an automated system and reduce the need for human resources to facilitate item tracking and handling. In order to fulfil all business logic, there will be a built in API system that allows third party developers to tailor the smart locker system to their specific business needs. This allows the system to be completely customizable, allowing certain businesses to custom tailor the system (hardware and software) to their specific business. The models described in this architecture document demonstrate the blueprint on how the smart locker system will be built.