Customer Support System

High Level Requirements

1. Project Overview

1.1 Scope

The purpose of this document is to describe the high level requirements for the Customer Support ticketing system. This document provides the objectives that the system will achieve by the end of the project cycle.

1.2 Audience

This high-level requirements document is intended to be used by members of the project team that will implement and verify the correct functionality of the system.

1.3 Objectives

- To provide customers with a new medium to resolve their issues.
- To provide management department with a new medium to manage and analyze results of the support team.

2. Project Requirements

2.1 Product Functionality

- The system shall allow the user the ability to signup and authenticate normally.
- The system shall allow the customer the ability to submit support requests.
- The system shall allow the customer the ability to view status of previous requests.
- The system shall allow the support agent the ability to find support requests.
- The system shall allow the support agent the ability to process support requests.
- The system shall allow the support agent the ability to view and print a report with all tickets closed in the last one month (should be PDF exportable).

- The system shall allow the admin the ability to delete a support request if it is possible.
- The system shall allow the admin the ability to delete a user if it is possible.

2.2 User Classes and Characteristics

There will be three user classes for the system.

- Admin
- Support Agent
- Customer

2.3 Operating Environment

- The Customer Support System server shall operate on a computer that is running Linux and has MySQL and Elasticsearch installed.
- The Customer Support System shall operate with Chrome and Safari browsers.

3. External Interface Requirements

3.1 User Interfaces

The user interface of the Customer Support System will be implemented in a web application format. The client will therefore be accessible through any Chrome and Safari web browsers.

3.2 Software Interfaces

The Customer Support System provides a API for other systems to access the system on the actual computer that is running the server.

4. Non-Functional Requirements

4.1 Performance

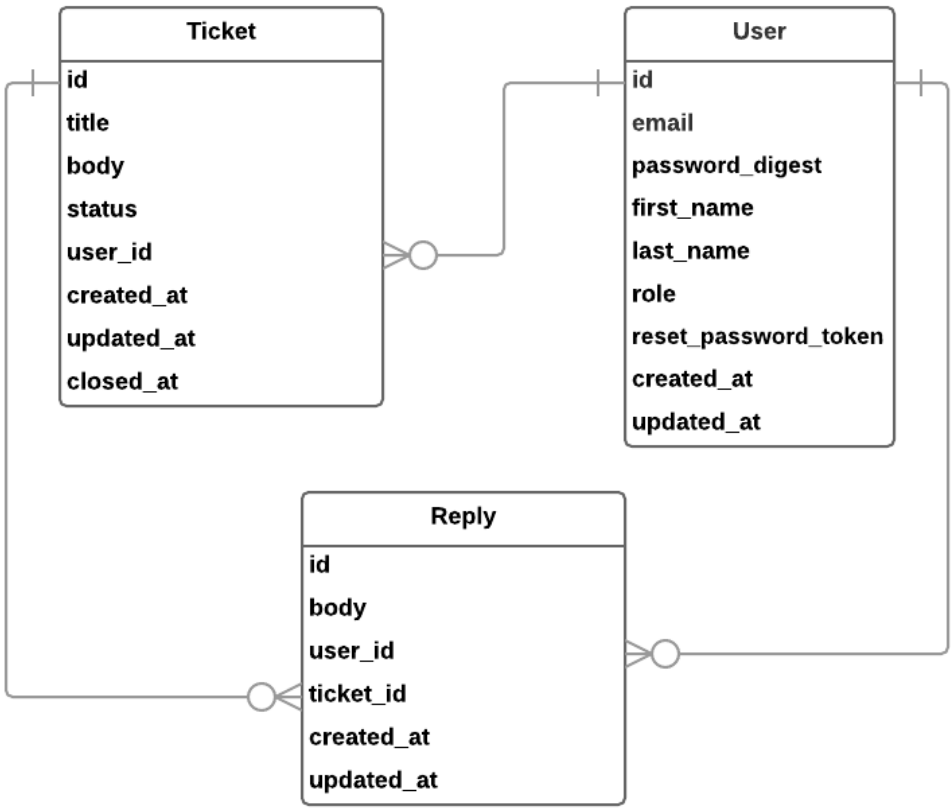
- Functions shall be performed in a timely manner.

4.2 Security

- The system shall use secure authentication.
- The system shall prevent any illegal access to the data.

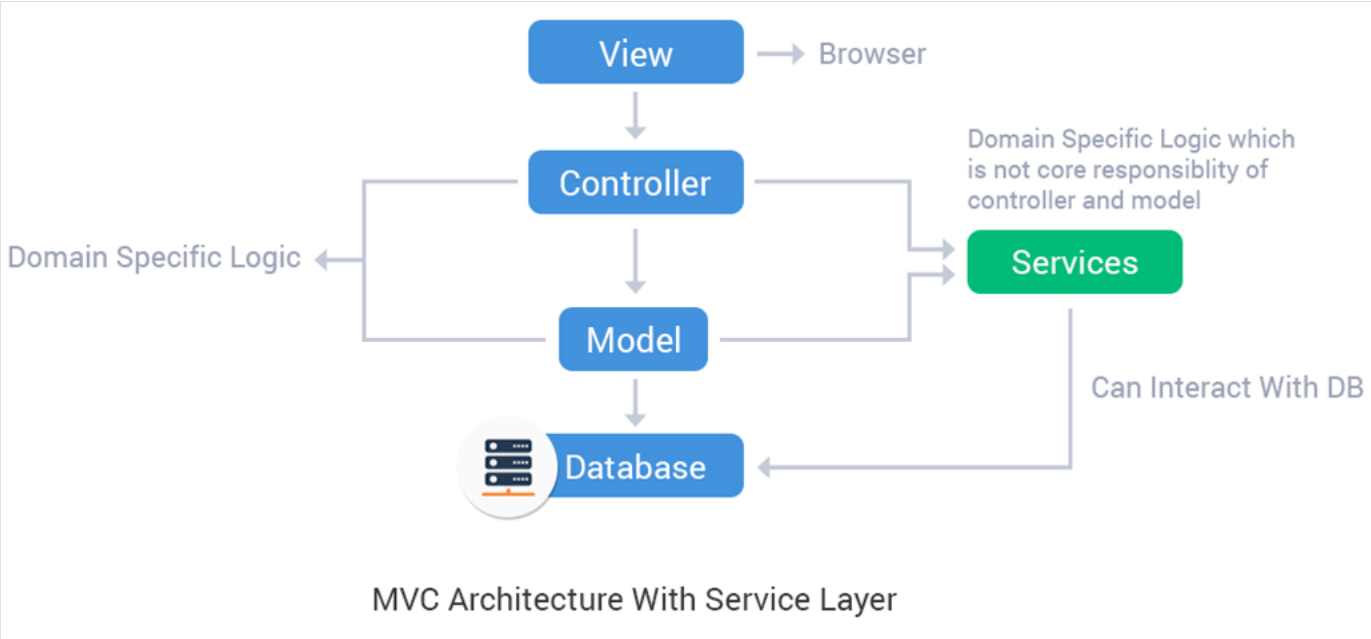
High level presentation of the data model

CUSTOMER SUPPORT ERD



Architecture

Backend (API)



MVC Architecture With Service Layer

It is a standard rails application with MVC architecture but with Service layer. Use of services helps in writing thin controllers as it enlightens a systematic way for keeping reused logic in a single place.

There are three services:

- `PasswordManager` - is responsible for sending and reseting passwords
- `TicketManager` - is responsible for creating, destroying, changing statuses and replying to tickets
- `UserManager` - is responsible for creating users and destroying users

When the managers become big it's better to extract some functions to a service object or new manager and call them from main manager. E.g. extract `TickerManager#reply` to `ReplyManager#create`:

```
class TickerManager
  def reply(ticket, reply, status)
    ReplyManager.new(@current_user).create(ticket, reply, status)
  end
end
```

or call a service object

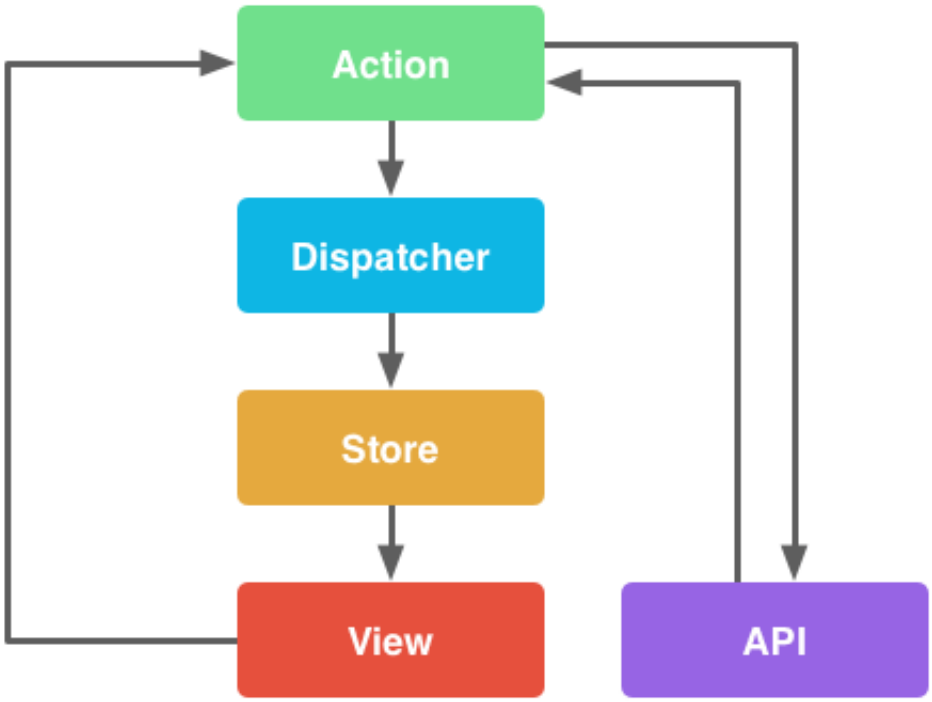
```
class TickerManager
  def reply(ticket, reply, status)
    # ...
```

```
    NotificationsService.notify(:reply, reply, @current_user)
    # ...
  end
end
```

Frontend (SPA)

It is a react-redux application.

Overview



There are four main components in a react-redux application.

- View - is a `React.Component` or just a function which is returning JSX
- Action - is a function which return a plain object with `type` property. It can also return a function with `redux-thunk` package.
- Store - a Redux store that holds the state tree. The only way to change the data in the store is to call `dispatch()` on it.
- Reducer - is a pure function which returns a new state. Reducers must not mutate previous state and must return a new object, instead. E.g.

  ```
  (state, action) => state.filter((ticket) => ticket.id !== action.id)
  ```

Also we have 7 routes:

- Home
  - Login
  - Signup
- Tickets
  - Ticket
- Report
- Users

These routes, pretty much, reflects the menu structure.

Other Diagrams

SUPPORT PROCESS