

Project

Mobile Tower Maintenance

15.07.2021

—

Sneha Saxena

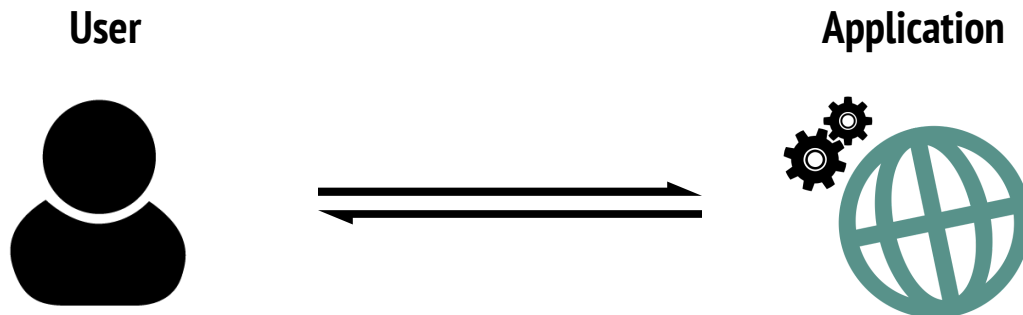
Overview

This is an initial documentation to design and develop a Mobile Tower Maintenance project that would provide the associated technicians and their corporate system to function and monitor in a structured way. This report consists information about following topics :-

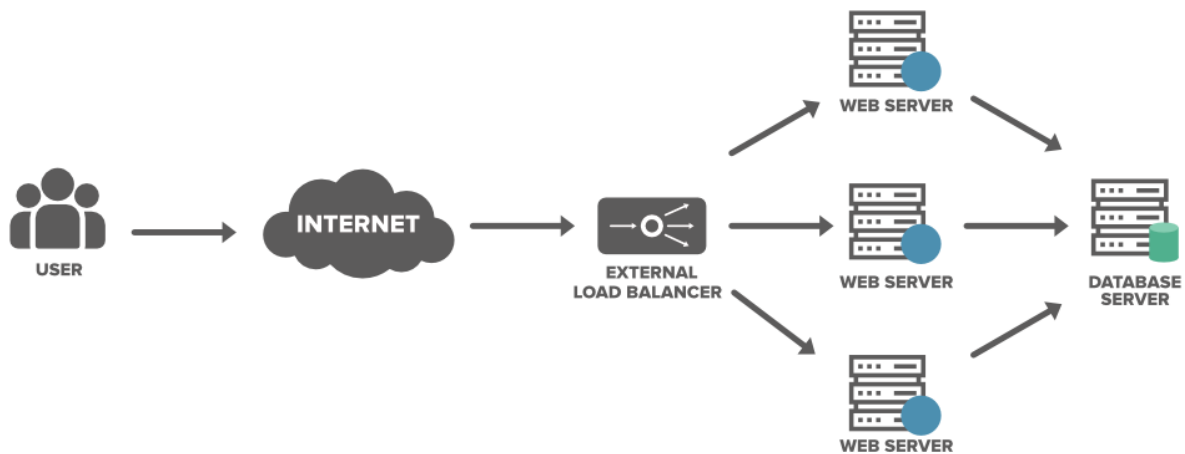
1. Architectural Representation
2. Requirement Specifications
3. Workflow
4. Features
5. Low Level Design

Architectural Representation

This section will introduce the initial architecture of the Application.




The main idea of this application is that technicians could connect to the system and perform various tasks required. To achieve this the first requirement that comes up is our application architecture.



User: User will interact with our web page/application which will make requests to the server via the internet.

Internet: Each request can travel back and forth via the Internet to provide response to the user.

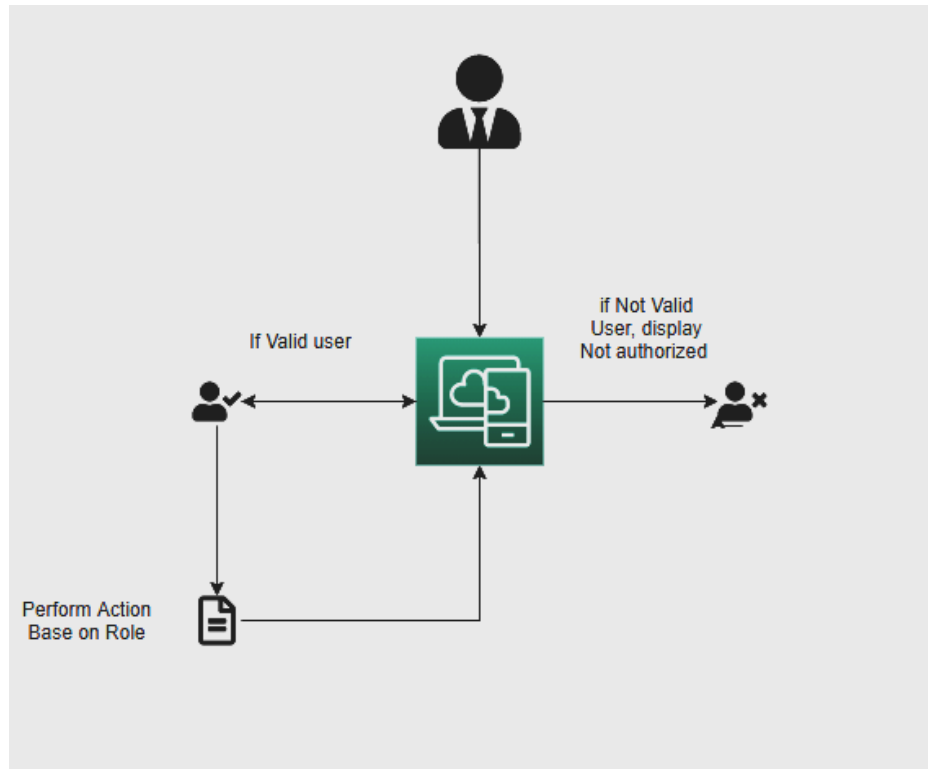


Load Balancer: Taking into consideration, there can be a huge amount of requests coming from users to our system, we need to create our system such that It could scale up and server the response to the request in an efficient manner. We will be placing a load balancer that would take care of such scenarios. For example we have millions of requests coming up per second, the load balancer will take care to direct them through the right instance keeping load distribution in thought. This will also handle such cases when a specific server goes down.

Web Servers: Web Servers are multiple instances of a specific service which will be created to process requests from users.

Database: We will be having two instances of the database for reading and writing operations respectively. One instance of the database will be in read only state that would serve the get request and logging the step in the database. Another instance of the database will be in a write state that would serve the post/pull and other operations in the database. This will help in keeping the system error prone. We will also implement a replication and trigger system that would keep both the instances in sync on runtime bases. This will also create a back-up of database for worst case scenarios.

WORKFLOW



Requirement Specifications

1. **Development tool:** Visual Studio, SQL Server Management Studio
2. **Technologies:**
 - a. C# Programming Language - To develop the core application.
 - b. .(Dot) Net Framework
 - c. Entity Framework
3. **Database:** SQL Server - To store/retrieve data.
4. **Server/Load Balancer** - AWS - To host service.
5. **Device/Internet** - To access the system.

Feature

User Authentication

1. Authentication :- An admin to the application will take care of users registration, assign them specific roles like technician and allow them to use the application. There will be a specific window for admin privileges. Users added will only be authorised to access the system.
2. Login :- Once the user is authorised, he will be able to login to our application with provided credentials by admin and the flexibility to change them in future via system. Only specific features will be visible to the users based on their role.

Privileges to Technician Role

Any maintenance request that comes up in the system, will be visible to all the users assigned with the "Technician" role.

1. They will have the advantage to even open any related maintenance requests.
2. Further, once a task is completed, they will be allowed to close the request.
3. In the meantime, due to any specific reasons justified for business, they will also have the right to reassign tasks to any other associated user.
4. They will have the flexibility to create new alerts, which could be sent to concerned users.
5. They will receive notifications during any action performed on their assigned task or related to any specific work.
6. Will have a specific window, where they could log their efforts spent over a task. Any assigned task to them, will already be visible to them, over the window. This is going to contribute to the generation of reports, which they can export and keep for a reference.
7. If required, the application will provide the feature to directly create a report and send it to seniors for evaluation or tracking purpose. This report could be generated on a filter basis as well, for eg: time duration.

Low Level Design

Model Layer

User Entities

1. Name (*Full Name*)
2. Unique ID
3. Managers name
4. Role
5. Department
6. Location
7. Mail Address

Database DAO (*communication with Database*)

1. Read user details
2. Perform authorization
3. Read/Write open Issues
4. Read/Write

Service Layer

Intermediate Between Model and Controller

1. Method for reading user details
2. Method for Viewing open Issues
3. Method for Perform Action to open Issues

Controller Layer


First to interact with web request (user's requests) and provide information interacting with service layer

1. Login Controller (*Login/Logout/Authorization/Validation*)
2. Admin Controller (*Writing/Master Privilege/Registrering/Override functionality*)
3. User Controller (*Read/write basic details/credential update/ other ..*)
4. Details Controller (*Perform all other feature communications*)

View Layer

Web front to provide User Interface UI to interact with applications.

1. Login/Logout Page
2. Admin Page

- 
3. User Information Page
 4. User Information Update Page
 5. Dashboard (*all action, pending request, notification and other*)
 6. Open Issues Pages (*Read/Write/Take Action*)
 7. Notification Page
-
- Model Layer is for database communication
 - View layer for front end
 - Controller is for communication between model and view