# Introduction

Information technology and software development have seen rapid growth like never before. Software systems were developed to automate tasks that were otherwise very tedious and difficult to carry out. The corporate sector had seen automation systems for almost every division they had. Technology turned out to be a boon for them.

But small business owners were not well supported in this rapid phase of technology until recent times. Mobile applications became a blessing for them too. But there are still certain sections which are yet to be automated. One such section is restaurant booking. Though this is available in developed nations, still a majority of the population are living in the cycle of traditional methods.

The project developed is a restaurant booking system which is distributed at its core allowing for multiple users of varying types and purposes have access to the same functionality.

# System Description

## Problems with existing system

The current demographic do not yet have a restaurant booking system in place. When one wants to eat at a restaurant booking a table is not possible unless it is a multiple star hotel wherein a call would suffice. Banking on sheer luck to get a table for a special occasion can be frustrating. Hence there is a problem which can be solved.

Though there are several solutions for a restaurant booking system, practical applications in the local market is not seen. The need for a better system which is simple enough to bring about market wide implementation is the motivation for this distributed restaurant booking system.

## The Proposed system

The proposed system consists of several components. These include an application each for desktop, web and mobile and a distributed backend functionality in the form of rest API. The three client applications are intended for different users and functionalities.

### Functionality of the system

The client applications are built for different use cases and so each one of them serve a different kind of user.

The mobile application allows a restaurant customer to view available tables and make reservations. The various types of tables in the restaurant are listed out with their availability. Once a customer wishes to reserve a particular table which is available, he can select the date and time and confirm the reservation. The application also allows the customer to cancel a reservation.

The web application serves the restaurant manager who deals with the customers and their reservations directly. The application which is responsive to adapt to any device the manager might be using, allows him to manage table listings as well as the table reservations.

The desktop application is intended for an admin who with the most elevated privileges can oversee all the functionalities and override on any of them in case of malfunction.

### Scope of the system

The system developed is intended for a particular restaurant chain and manages the complete table booking process. All the necessary features required for the customer as well the restaurant employees to participate in the process is included.

The table booking process briefly includes:

* Customer can make a new reservation or cancel an existing one
* Restaurant manager manages the table listing as well as the reservations made by a customer
* An admin can oversee all the reservation and table listing process. He also has rights to make update any of them in case of emergency

# Background Research

A distributed application with context to a web platform usually consists of applications spanning over multiple computers or systems. These applications share resources and each of them handle a different process. A standard architecture used to consist of a client and server wherein a client used to access functionalities from a server which usually was composed of several machines.

With *middleware* introduced to software architecture, different ways of implementation came up. Traditionally used methods were Remote Method Invocation (RMI) a popular Java API, Common Object Request Broker Architecture (CORBA) a standard for communication among systems and Simple Object Access Protocol (SOAP) which was another popular application protocol for handling messages.

Recent times saw the rise of the REST architecture which stands for Representational State Transfer. It is a stateless architecture which is works with the HTTP protocol. Implementing REST services allows for convenient development over multiple platforms and makes for easy data handling as well. REST architecture distributes its state and functionality over resources and each of these can be handled with a set of commands for the HTTP verbs such as GET, POST, PUT and DELETE.

The project developed makes use of the REST architecture to handle the distributed backend functionalities.

# System Design

## Network Diagram

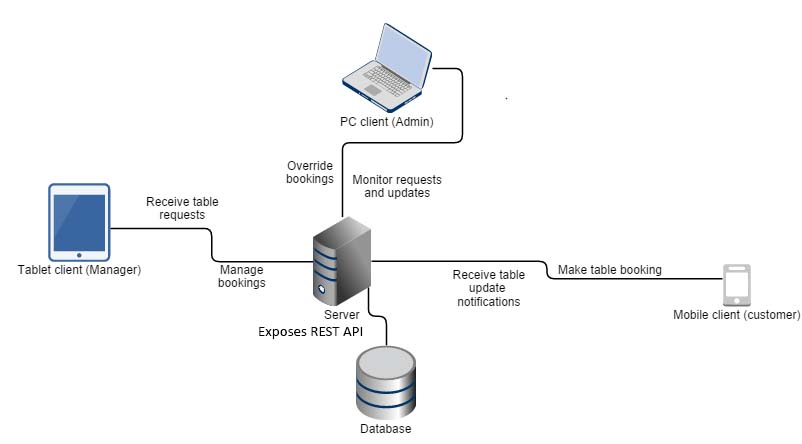


Figure 1 Network diagram of the System

## Use case Diagram

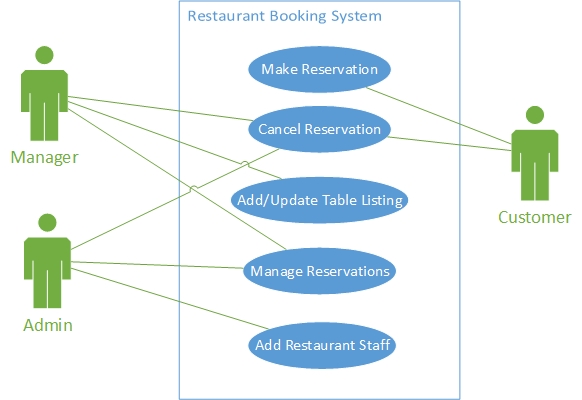


Figure 2 Use case diagram

## Entity Relationship Diagram

## Application interface designs

# Technical Description

The system consists of several components and applications and each of these are developed with different technologies. At the core, the system makes use of the REST architecture. Through the exposure of REST APIs the client applications are able to access data while the actual processing is handled elsewhere.

## Technologies used

A number of technologies play a role in the development of the system. Though the diverse nature of technologies used, the applications and components are able to work together due the distributed nature of the system architecture.

The RESTful APIs exposed through the backend is built with Node.js which is a JavaScript implementation for the server. It is assisted with Express.js which is web framework built on top of Node.js. This will serve as the web service communicating between the different client applications and the database. MySQL serves as the database for the system.

The mobile application is built using hybrid technologies which make use HTML, CSS and JavaScript. It makes use of the Ionic framework which is a popular mobile framework and compiles the application using Cordova, the popular hybrid framework. Ionic is built on top AngularJS, a JavaScript framework for building the front end of web applications.

// Web application comes here

// Desktop application comes here

## Implementation

### Backend and REST APIs

ExpressJS comes with a standard bootstrapped project structure with the basic codes included as default. Such a project was set up and the necessary components were added. ExpressJS provides a router mechanism to handle HTTP requests and responses. The use of this router makes it easy to setup handles for the various requests and URLs and hence expose the necessary APIs to the clients.

NodeJS provides the core functionality to run JavaScript on the server. Being an open sourced framework it has thousands of third party modules which can be added into the project to provide support for various functionalities. One such module was used to add the MySQL database support.

Route requests were handled in ExpressJS to allow requests be made using the basic GET, POST, PUT and DELETE requests on the data models. These handlers were set up for all the required functionalities for the client applications. They handle the backend and database processing and pass on the results to the client in JSON format. Additional handles can be set up if required.

### Mobile application

Ionic framework is a very popular modern hybrid framework which provides standard classes and components for the various elements found on the native mobile application of iOS and Android. It also provided a few starter templates which consists of the basic structure to run a simple mobile application. Such a starter template was set up and the required pages and components was added.

Ionic provides a MVC architecture to the application and hence makes it easy to separate the code and functionalities. Every page has a controller which handles the required HTTP calls to the appropriate web API in order to access data and update the view or the page.

### Web application

### Desktop application

## Additional dependencies

The core of the distributed system developed is built using Node.js and hence requires an environment which supports it. When running on a local machine it requires Node.js to be installed on the machine and when hosted on an online server or hosting space it needs one which has support for it.

The mobile application uses Ionic, a hybrid framework, but can be compiled to either android or iOS compatible applications. For this it needs the development kit or SDKs of the respective mobile platform to compile the hybrid application code.

# Problems faced

Developing a distributed system has its fair share of issues and obstacles but this project had a few of them as well. These were mainly in the form of technical issues and time management.

Three client applications were developed and these were on different platforms using different technologies. In addition, the REST API was built in another language. Hence there was the issue of handling various languages and bug fixes for all of them. But these were relatively easy to overcome using the technology documentations and other online guides.

Being a team project there was another concern of managing the code base among the members. But this was easily sorted out by using a Git repository on Bitbucket. Timely commits and regular communications among the members smoothened the development process.

The biggest issue faced was with time management. Having to handle other projects on a similar timeline caused for breaking down of work among the various commitments. This also had an effect on the attention and focus given to this project. Though difficult, progress was made and development was complete within the deadline.

# Future prospects

The restaurant booking system developed had limited scope due to the nature of the project and the project timeline. Certain features and user experience aspects which would otherwise be in a standard booking system were not included for the same reason. These however could be further additions to the project in order to give a more holistic experience for the booking system.

## Commercial expansion

The restaurant booking system was developed for a particular restaurant chain. But these could easily be expanded so as to give the same functionality and backend code base for other restaurants. Thus there is a larger scope and commercial value in the project.

# Summary

A restaurant booking system was developed which supported multiple client applications in the form of mobile, web and desktop. These clients communicate to a backend server which exposes data and functionalities through REST APIs.

This distributed architecture of the system allows for better resource allocation and performance of the applications involved. Also since the web API used follow the HTTP protocols it is possible to expand the application to provide additional features to the client applications.

# Bibliography

* Barry, D.K. (2000) Semantic vocabularies. Available at: http://www.service-architecture.com/articles/web-services/representational\_state\_transfer\_rest.html (Accessed: 19 March 2016).
* Fielding, R.T. (2016) Fielding Dissertation: CHAPTER 5: Representational state transfer (REST). Available at: https://www.ics.uci.edu/~fielding/pubs/dissertation/rest\_arch\_style.htm (Accessed: 20 March 2016).
* Matsudaira, K. (2016) The architecture of open source applications (volume 2): Scalable web architecture and distributed systems. Available at: http://www.aosabook.org/en/distsys.html (Accessed: 21 March 2016).
* Perera, S. (2011) Developing Distributed Web Applications, Where does REST fit in? Available at: http://www.slideshare.net/hemapani/developing-distributed-web-applications-where-does-rest-fit-in-8083222 (Accessed: 19 March 2016).