

A Project Review-2 Report  
on  
CLUB EVENT REGULATOR AND  
UPDATER

Submitted in partial fulfillment of the  
requirement for the award of the degree of

Bachelor of Technology  
In  
Computer Science



Under The  
Supervision of Dr.M.  
Thirunavukkarasan  
Assistant Professor

Submitted By

Twinkle Deep  
18SCSE1010655  
Shubham Kashyap  
18SCSE1010061

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING  
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING  
GALGOTIAS UNIVERSITY, GREATER NOIDA  
INDIA  
OCTOBER, 2021



**SCHOOL OF COMPUTING SCIENCE AND  
ENGINEERING  
GALGOTIAS UNIVERSITY, GREATER NOIDA**

**CANDIDATE'S DECLARATION**

We hereby certify that the work which is being presented in the thesis/project/dissertation, entitled "**CLUB EVENT REGULATOR AND UPDATER**" in partial fulfillment of the requirements for the award of the Bachelor of Technology In Computer Science submitted in the School of Computing Science and Engineering of Galgotias University, Greater Noida, is an original work carried out during the period of September, 2021 to May, 2021, under the supervision of Dr.M. Thirunavukkarasan Assistant Professor, Department of Computer Science and Engineering, of School of Computing Science and Engineering , Galgotias University, Greater Noida

The matter presented in the project has not been submitted by me/us for the award of any other degree of this or any other places.

Twinkle Deep 18SCSE1010655  
Shubham Kashyap 18SCSE1010061

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Supervisor  
Name  
Designation

**CERTIFICATE**

The Final Project Viva-Voce examination of Twinkle Deep (18SCSE1010655) and Shubham Kashyap (18SCSE1010061) has been held on \_\_\_\_\_ and their work is recommended for the award of Bachelor of Technology in Computer Science.

**Signature of Examiner(s)**

**Signature of  
Supervisor(s)**

**Signature of Project Coordinator**

**Signature of  
Dean**

## Table of Contents

Title	Page No.
Acknowledgement	4
Abstract	5
Chapter 1 Introduction	6
Chapter 2 Literature Survey	11
Chapter 3 Project Design	13
Chapter 4 Snapshots of Design	15
Chapter 5 Module Description	17
Chapter 6 Result and Discussions	55
Chapter 7 Conclusion	56
Chapter 8 References	57

## **Acknowledgement**

We wish to express our heartfelt gratitude to all the people who have played a crucial role in the research for this project, without their active cooperation the preparation of this project could not have been proceeding in the way it is proceeding at the current given time.

We are thankful to our respected Guide, Dr.M. Thirunavukkarasan, Assistant Professor, for motivating us to complete this project with complete focus and attention.

He has always continued to guide us through all the problems we have faced in the process of making this project from scratch.

We are also thankful to our reviewers who supported us and made us realize the shortcomings in our project, all this time, with utmost cooperation and patience, helping us in advancing with this project of ours.

## **Abstract**

Every college has their own set of clubs and it's a task to maintain records and make each announcement through emails. The problem with E-mail announcements is that it does not provide any surety that every person has been informed and the process is very cumbersome and boring for the masses.

Therefore, using this web based MERN project we aim to make a website that not only maintains but also provides for attractive and new ways to let the masses know about club events and even actively participate in them. It'll have different logins for club administrators and club members, providing them with different administrative tools.

The tools and technology used would be:

- MongoDB
- Express
- React
- Node

By this project we aim to provide a revised and seamless platform for maintenance and updating of clubs' events in the respective institutions.

## Introduction

The technology stack is a set of frameworks and tools used to develop a software product. This set of frameworks and tools are very specifically chosen to work together in creating a well-functioning software.

Here are some examples of widely used web development technology stacks in use today:

MERN (MongoDB, ExpressJS, ReactJS, NodeJS)

LAMP (Linux, Apache, MySQL, PHP)

MEAN (MongoDB, ExpressJS, AngularJS, NodeJS)

MERN stack is a web development framework. It consists of MongoDB, ExpressJS, ReactJS, and NodeJS as its working components. Here are the details of what each of these components is used for in developing a web application when using MERN stack:

**MongoDB:** A document-oriented, No-SQL database used to store the application data.

**NodeJS:** The JavaScript runtime environment. It is used to run JavaScript on a machine rather than in a browser.

**ExpressJS:** A framework layered on top of NodeJS, used to build the backend of a site using NodeJS functions and structures. Since NodeJS was not developed to make websites but rather run JavaScript on a machine, ExpressJS was developed.

ReactJS: A library created by Facebook. It is used to build UI components that create the user interface of the single page web application.



*Figure 1*

As shown in the illustration above, the user interacts with the ReactJS UI components at the application front-end residing in the browser. This frontend is served by the application backend residing in a server, through ExpressJS running on top of NodeJS.

Any interaction that causes a data change request is sent to the NodeJS based Express server, which grabs data from the MongoDB database if required, and returns the data to the frontend of the application, which is then presented to the user.

MERN stack is a technology that has been gaining a lot of popularity in the recent times due to its high level of efficiency in the field of full stack development.

Management of everything is moving towards digitization and using our project we are looking forward to digitizing the updating and regulation of the college clubs.

MERN is a user-friendly full-stack JavaScript framework ideal for building dynamic websites and applications. It is a free and open-source stack designed to supply developers with a quick and organized method for creating rapid prototypes of MERN-based web applications.

One of the main benefits of the MERN stack is that a single language, JavaScript, runs on every level of the application, making it an efficient and modern approach to web development.

We look forward to developing a full stack project using MERN which will give the users two paths. The first one being the Administrator and second being the Member.

The Administrator and Members will have their own set of features available for them. The tools and technology used would be:

- MongoDB
- Express
- React
- Node

Using the implementation of this project we aim to provide a revised and seamless platform for maintenance and updating of clubs' events in the respective institutions.



It's hard to accomplish much on the web without JavaScript, which is the single language that runs the entire MERN full stack and boasts one of the most active developer communities. Because every part of MERN programming is written in one language, it allows unique server-side and client-side execution environments. Valued for its versatility in building fast, robust and maintainable production web applications, MERN is in high demand with numerous startups and employers.

Advantages of MERN Stack:

UI rendering and performance

React JS is the best when it is about UI layer abstraction. Since React is only a library, it provides you the freedom to build the application and organize the code however you want. So, it is better than Angular in terms of UI rendering and performance.

2. Cost-Effective

As MERN Stack uses one language throughout that is Javascript so it will be beneficial for a company to hire Javascript experts only rather than hiring different specialists for different technology. This move will save a lot of time and money.

3. Open Source

All technologies that are involved in MERN are open-source. This feature allows a developer to get solutions to queries that may evolve during development, from the available open portals. As a result, it will be beneficial for a developer.

#### 4. Easy to switch between client and server

As everything is written in one language this is why MERN is simple and fast.

And also it is easy to switch between client and server.

### Disadvantages of MERN Stack

#### 1. Productivity

Since React is just a library it uses many third-party libraries which provides lower developer productivity. And due to this upgrading, the React code requires more effort.

#### 2. Large-Scale Applications

It becomes difficult with MERN to make a large project in which many developers are involved. MERN stack is best suited for single-page applications.

#### 3. Error prevention:

If you want a technology stack that prevents common coding errors by its very design, then the MEAN stack is a better choice. As Angular uses Typescript, so prevents common coding errors at the coding stage itself. However, React lags behind here.

## **Literature Review**

MERN stack is becoming a highly popular technology and recently there have been a lot of research papers available to read and learn new knowledge from. One of the papers from IJERT, on, “Performance Optimization using MERN stack on Web Applications”, helps us know about how using the knowledge of MERN stack we can optimize the performance of the websites we design.

MERN stack is becoming a highly popular technology and recently there have been a lot of research papers available to read and learn new knowledge from. One of the papers from IJERT, on, “Performance Optimization using MERN stack on Web Applications”, helps us know about how using the knowledge of MERN MERN stack is becoming a highly popular technology and recently there have been a lot of research papers available to read and learn new knowledge from.

One of the papers from IJERT, on, “Performance Optimization using MERN stack on Web Applications”, helps us know about how using the knowledge of MERN stack we can optimize the performance of the websites we design. Reading another paper on “Content Management on Websites” gave us the knowledge about how we can efficiently manage contents on a website we design. The paper also stated the need of portals and how important they are to us.

The paper also gave information about the architecture of the portals and about how they can be made more secure. We tend to read more such papers related to our project and use all the information gained from them, in our

project. Using MERN stack we can optimize the performance of the websites we design.

Reading another paper on “Content Management on Websites” gave us the knowledge about how we can efficiently manage contents on a website we design. The paper also stated the need of portals and how important they are to us. The paper also gave information about the architecture of the portals and about how they can be made more secure.

We tend to read more such papers related to our project and use all the information gained from them, in our project.

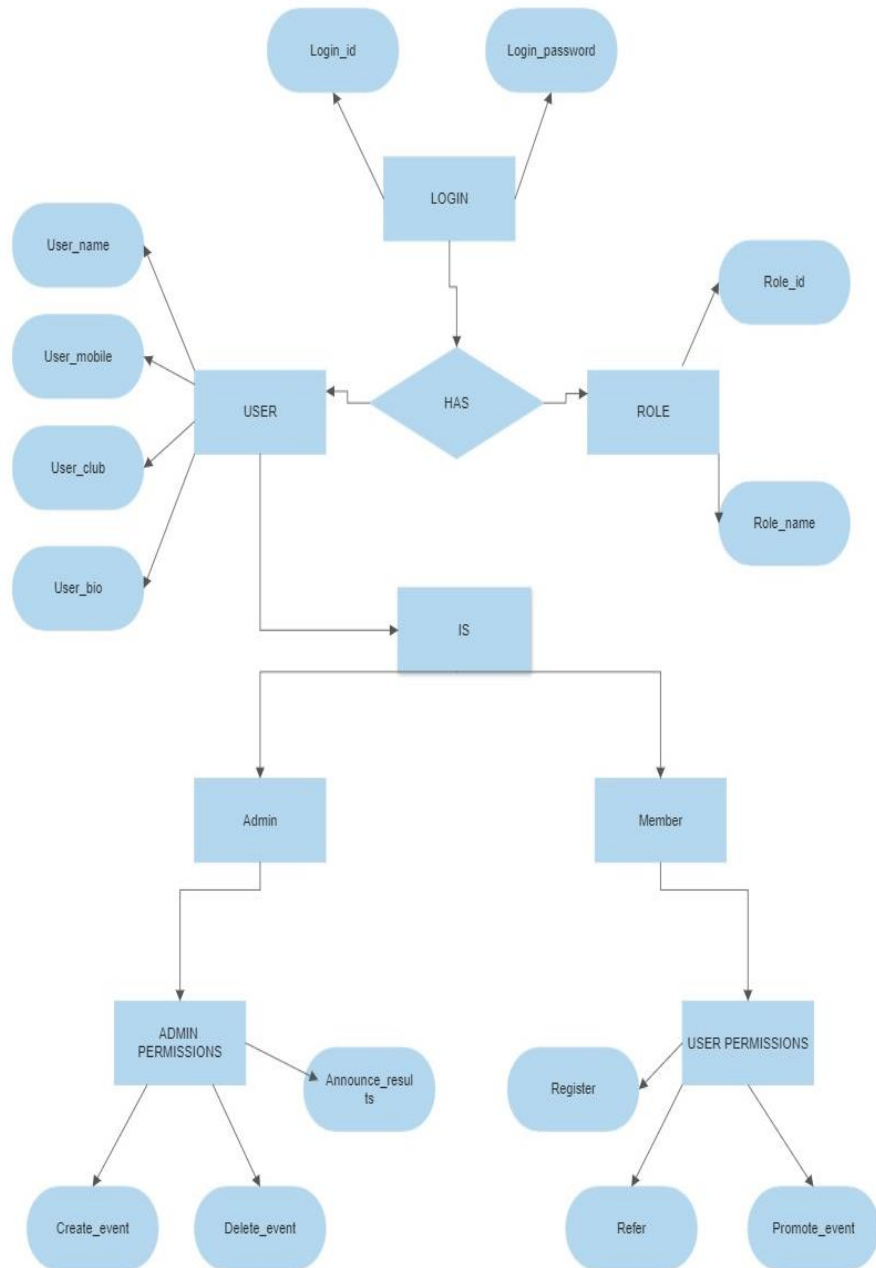
Reading another paper on “Content Management on Websites” gave us the knowledge about how we can efficiently manage contents on a website we design. The paper also stated the need of portals and how important they are to us.

The paper also gave information about the architecture of the portals and about how they can be made more secure.

We tend to read more such papers related to our project and use all the information gained from them, in our project.

# Project Design

## ER-Diagram:



**Functionality for end user:**

- Log In
- Refer
- Promote Event
- Check Upcoming Events of Schools and clubs
- Register for events
- Check profiles of event organizer

**Functionality for Schools and clubs' admin:**


- Log In
- Create events
- Delete events
- Announce Results
- See registered student
- Manage your profile

**Basic working:**

Students have to login using their college credentials. Then they can select any school or club. On the next screen student can see all the upcoming events and auditions of respective school or club. Student will get all the details of event/audition and can register for them. Students can also check the profile of the event or audition organizers. Below are the few images to understand it better.

## Snapshots of the Design

### Add Your Education

 Add any school, bootcamp, etc that you have attended

\* - required field

\* School or Bootcamp

\* Degree or Certificate

Field Of Study

From Date

dd-mm-yyyy

Current School or Bootcamp

To Date


dd-mm-yyyy

Program Description

Submit      Go Back

Figure 1

### Create Your Profile

 Let's get some information to make your profile stand out

\* - required field

\* Select Professional Status

Give us an idea of where you are at in your career

Company

Could be your own company or one you work for

Website

Could be your own or a company website

Location

City & state suggested (eg. Boston, MA)

\* Skills

Please use comma separated values (eg. HTML,CSS,JavaScript,PHP)

Github Username

If you want your latest repos and a Github link, include your username

A short bio of yourself

Tell us a little about yourself

Add Social Network Links:      Optional


 Twitter URL

Figure 2

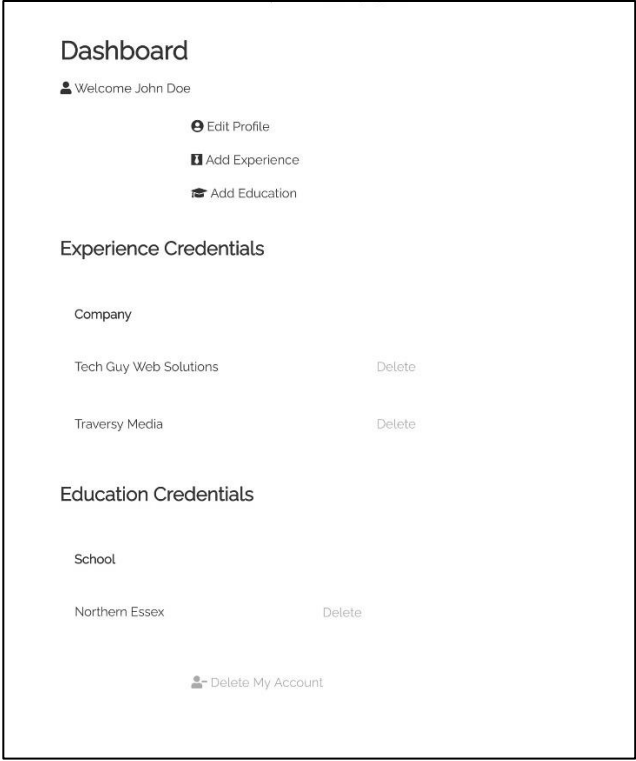


Figure 3

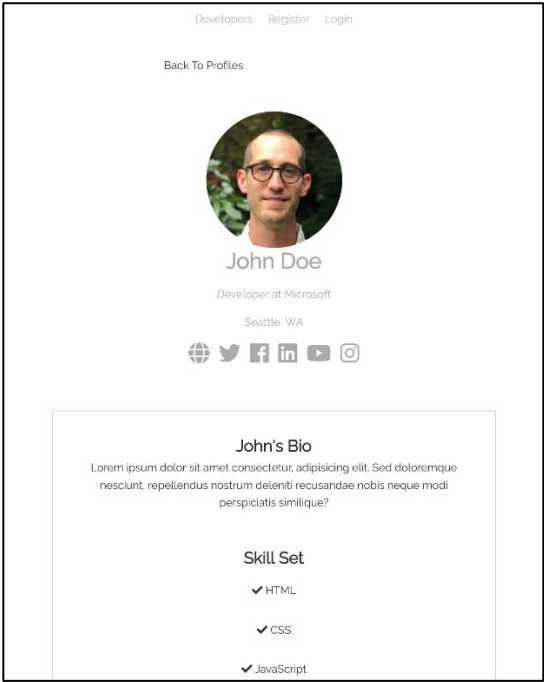


Figure 4



## Modules Description

### MongoDB

#### Introduction to MongoDB

Welcome to the MongoDB 5.0 Manual! MongoDB is a document database designed for ease of development and scaling. The Manual introduces key concepts in MongoDB, presents the query language, and provides operational and administrative considerations and procedures as well as a comprehensive reference section.

MongoDB offers both local and cloud-hosted deployment options:

- For locally hosted deployments, MongoDB offers both a *Community* and an *Enterprise* version of the database:
  - MongoDB Community is the source available and free to use edition of MongoDB.
  - MongoDB Enterprise is available as part of the MongoDB Enterprise Advanced subscription and includes comprehensive support for your MongoDB deployment. MongoDB Enterprise also adds enterprise-focused features such as LDAP and Kerberos support, on-disk encryption, and auditing.
- MongoDB Atlas is a hosted MongoDB Enterprise service option in the cloud which requires no installation overhead and offers a free tier to get started.

#### Document Database

A record in MongoDB is a document, which is a data structure composed of field and value pairs. MongoDB documents are similar to JSON objects. The values of fields may include other documents, arrays, and arrays of documents.

The advantages of using documents are:

- Documents (i.e. objects) correspond to native data types in many programming languages.
- Embedded documents and arrays reduce need for expensive joins.
- Dynamic schema supports fluent polymorphism.

### **Collections/Views/On-Demand Materialized Views**

MongoDB stores documents in collections. Collections are analogous to tables in relational databases.

In addition to collections, MongoDB supports:

- Read-only Views (Starting in MongoDB 3.4)
- On-Demand Materialized Views (Starting in MongoDB 4.2).

### **Key Features**

#### **High Performance**

MongoDB provides high performance data persistence. In particular,

- Support for embedded data models reduces I/O activity on database system.
- Indexes support faster queries and can include keys from embedded documents and arrays.

### **Rich Query Language**

MongoDB supports a rich query language to support read and write operations (CRUD) as well as:

- Data Aggregation
- Text Search and Geospatial Queries.

### **See also:**

- [SQL to MongoDB Mapping Chart](#)
- [SQL to Aggregation Mapping Chart](#)
- [Learn about the latest query language features with the MongoDB Query Language: What's New presentation from \*\*MongoDB.live 2020\*\*.](#)

### **High Availability**

MongoDB's replication facility, called replica set, provides:

- *automatic* failover
- data redundancy.

A replica set is a group of MongoDB servers that maintain the same data set, providing redundancy and increasing data availability.

### **Horizontal Scalability**

MongoDB provides horizontal scalability as part of its *core* functionality:

- Sharding distributes data across a cluster of machines.
- Starting in 3.4, MongoDB supports creating zones of data based on the shard key. In a balanced cluster, MongoDB directs reads and writes covered by a zone only to those shards inside the zone. See the Zones manual page for more information.

### **Support for Multiple Storage Engines**

MongoDB supports multiple storage engines:

- WiredTiger Storage Engine (including support for Encryption at Rest)
- In-Memory Storage Engine.

In addition, MongoDB provides pluggable storage engine API that allows third parties to develop storage engines for MongoDB.

### **Install MongoDB Community Edition on Windows**

#### **MongoDB Atlas**

MongoDB Atlas is a hosted MongoDB service option in the cloud which requires no installation overhead and offers a free tier to get started.

#### *1. Overview*

Use this tutorial to install MongoDB 5.0 Community Edition on Windows using the default installation wizard.

## 1. MongoDB Version

This tutorial installs MongoDB 5.0 Community Edition. To install a different version of MongoDB Community, use the version drop-down menu in the upper-left corner of this page to select the documentation for that version.

## 2. Installation Method

This tutorial installs MongoDB on Windows using the default MSI installation wizard. To install MongoDB using the `msiexec.exe` command-line tool instead, see Install MongoDB using msiexec.exe.

The `msiexec.exe` tool is useful for system administrators who wish to deploy MongoDB in an unattended fashion using automation.

### 2. Considerations

#### 1. MongoDB Shell, `mongosh`

The MongoDB Shell (`mongosh`) is not installed with MongoDB Server. You need to follow the `mongosh` installation instructions to download and install `mongosh` separately.

#### 2. Platform Support

MongoDB 5.0 Community Edition supports the following **64-bit** versions of Windows on x86\_64 architecture:

- Windows Server 2019
- Windows 10 / Windows Server 2016

MongoDB only supports the 64-bit versions of these platforms.

See Supported Platforms for more information.

### **3. Virtualization**

Oracle offers experimental support for VirtualBox on Windows hosts where Hyper-V is running. However, Microsoft does not support VirtualBox on Hyper-V.

Disable Hyper-V if you want to install MongoDB on Windows using VirtualBox.

### **4. Production Notes**

Before deploying MongoDB in a production environment, consider the Production Notes document which offers performance considerations and configuration recommendations for production MongoDB deployments.

#### *3. Install MongoDB Community Edition*

##### **1. Procedure**

Follow these steps to install MongoDB Community Edition using the MongoDB Installer wizard. The installation process installs both the MongoDB binaries as well as the default configuration file <install directory>\bin\mongod.cfg.

##### **1. Download the installer.**

Download the MongoDB Community [.msi](#) installer from the following link:

► MongoDB Download Center

- a. In the **Version** dropdown, select the version of MongoDB to download.
- b. In the **Platform** dropdown, select **Windows**.
- c. In the **Package** dropdown, select **msi**.
- d. Click **Download**.

**2. Run the MongoDB installer.**

For example, from the Windows Explorer/File Explorer:

- a. Go to the directory where you downloaded the MongoDB installer (.msi file). By default, this is your **Downloads** directory.
- b. Double-click the .msi file.

**3. Follow the MongoDB Community Edition installation wizard.**

The wizard steps you through the installation of MongoDB and MongoDB Compass.

**a. Choose Setup Type**

You can choose either the **Complete** (recommended for most users) or **Custom** setup type. The **Complete** setup option installs MongoDB and the MongoDB tools to the default location. The **Custom** setup option allows you to specify which executables are installed and where.

**b. Service Configuration**

Starting in MongoDB 4.0, you can set up MongoDB as a Windows service during the install or just install the binaries.

## MongoDB ServiceMongoDB

The following installs and configures MongoDB as a Windows service.

Starting in MongoDB 4.0, you can configure and start MongoDB as a Windows service during the install, and the MongoDB service is started upon successful installation.

- Select **Install MongoDB as a Service** MongoDB as a service.
- Select either:
  - **Run the service as Network Service user** (Default)  
This is a Windows user account that is built-in to Windows  
**or**
  - **Run the service as a local or domain user**
    - For an existing local user account, specify a period (i.e. `.`) for the **Account Domain** and specify the **Account Name** and the **Account Password** for the user.
    - For an existing domain user, specify the **Account Domain**, the **Account Name** and the **Account Password** for that user.
- **Service Name.** Specify the service name. Default name is `MongoDB`. If you already have a service with the specified name, you must choose another name.
- **Data Directory.** Specify the data directory, which corresponds to the `--dbpath`. If the directory does not exist, the installer will create the directory and sets the directory access to the service user.



- **Log Directory.** Specify the Log directory, which corresponds to the `--logpath`. If the directory does not exist, the installer will create the directory and sets the directory access to the service user.

**c. Install MongoDB Compass**

*Optional.* To have the wizard install MongoDB Compass, select **Install MongoDB Compass (Default)**.

- d. When ready, click **Install**.

**2. Install `mongosh`**

The `.msi` installer does not include `mongosh`. Follow the `mongosh` installation instructions to download and install the shell separately.

**3. If You Installed MongoDB as a Windows Service**

The MongoDB service starts upon successful installation. Configure the MongoDB instance with the configuration file `<install directory>\bin\mongod.cfg`.

**4. If You Did Not Install MongoDB as a Windows Service**

If you only installed the executables and did not install MongoDB as a Windows service, you must manually start the MongoDB instance.

See [Run MongoDB Community Edition from the Command Interpreter](#) for instructions to start a MongoDB instance.

#### 4. *Run MongoDB Community Edition as a Windows Service*

Starting in version 4.0, you can install and configure MongoDB as a **Windows Service** during installation. The MongoDB service starts upon successful installation. Configure the MongoDB instance with the configuration file <install directory>\bin\mongod.cfg.

If you have not already done so, follow the mongosh installation instructions to download and install the MongoDB Shell (mongosh).

Be sure to add the path to your `mongosh.exe` binary to your `PATH` environment variable during installation.

Open a new **Command Interpreter** and enter `mongosh.exe` to connect to MongoDB.

For more information on connecting to a `mongod` using `mongosh.exe`, such as connecting to a MongoDB instance running on a different host and/or port, see [Connect to a Deployment](#).

For information on CRUD (Create, Read, Update, Delete) operations, see:

- [Insert Documents](#)
- [Query Documents](#)
- [Update Documents](#)
- [Delete Documents](#)

## 1. Start MongoDB Community Edition as a Windows Service

To start/restart the MongoDB service, use the Services console:

1. From the Services console, locate the MongoDB service.
2. Right-click on the MongoDB service and click **Start**.

## 2. Stop MongoDB Community Edition as a Windows Service

To stop/pause the MongoDB service, use the Services console:

1. From the Services console, locate the MongoDB service.
2. Right-click on the MongoDB service and click **Stop** (or **Pause**).

## 3. Remove MongoDB Community Edition as a Windows Service

To remove the MongoDB service, first use the Services console to stop the service. Then open a Windows command prompt/interpreter (`cmd.exe`) as an **Administrator**, and run the following command:

```
sc.exe delete
```

```
MongoDB
```

## 5. *Run MongoDB Community Edition from the Command Interpreter*

You can run MongoDB Community Edition from the Windows command prompt/interpreter (`cmd.exe`) instead of as a service.

Open a Windows command prompt/interpreter (`cmd.exe`) as an **Administrator**.

You must open the command interpreter as an **Administrator**.

### 1. **Create database directory.**

Create the data directory where MongoDB stores data. MongoDB's default data directory path is the absolute path `\data\db` on the drive from which you start MongoDB.

From the **Command Interpreter**, create the data directories:

```
cd C:\  
md "\\data\db"
```

### 2. **Start your MongoDB database.**

To start MongoDB, run `exe`.

```
"C:\Program Files\MongoDB\Server\5.0\bin\mongod.exe" --  
dbpath="c:\data\db"
```

The `--dbpath` option points to your database directory.

If the MongoDB database server is running correctly, the **Command Interpreter** displays:

```
[initandlisten] waiting for connections
```

Depending on the Windows Defender Firewall settings on your Windows host, Windows may display a **Security Alert** dialog box about blocking "some features" of `C:\Program Files\MongoDB\Server\5.0\bin\mongod.exe` from communicating on networks. To remedy this issue:

- a. Click **Private Networks, such as my home or work network**.
- b. Click **Allow access**.

To learn more about security and MongoDB, see the Security Documentation.

### 3. Connect to MongoDB.

If you have not already done so, follow the mongosh installation instructions to download and install the MongoDB Shell (mongosh).

Be sure to add the path to your `mongosh.exe` binary to your `PATH` environment variable during installation.

Open a new **Command Interpreter** and enter `mongosh.exe` to connect to MongoDB.

For more information on connecting to `mongod` using `mongosh.exe`, such as connecting to a MongoDB instance running on a different host and/or port, see [Connect to a Deployment](#).

For information on CRUD (Create, Read, Update, Delete) operations, see:

- Insert Documents
- Query Documents
- Update Documents
- Delete Documents

## 6. *Additional Considerations*

### 1. **Localhost Binding by Default**

By default, MongoDB launches with `bindIp` set to `127.0.0.1`, which binds to the localhost network interface. This means that the `mongod.exe` can only accept connections from clients that are running on the same machine. Remote clients will not be able to connect to the `mongod.exe`, and the `mongod.exe` will not be able to initialize a replica set unless this value is set to a valid network interface.

This value can be configured either:

- in the MongoDB configuration file with `bindIp`, or
- via the command-line argument `--bind_ip`

Before binding to a non-localhost (e.g. publicly accessible) IP address, ensure you have secured your cluster from unauthorized access. For a complete list of security recommendations, see Security Checklist. At minimum, consider enabling authentication and hardening network infrastructure.

For more information on configuring `bindIp`, see IP Binding.

## 2. Point Releases and `.msi`

If you installed MongoDB with the Windows installer (`.msi`), the `.msi` automatically upgrades within its release series (e.g. 4.2.1 to 4.2.2).

Upgrading a full release series (e.g. 4.0 to 4.2) requires a new installation.

## 3. Add MongoDB binaries to the System PATH

If you add `C:\Program Files\MongoDB\Server\5.0\bin` to your System `PATH` you can omit the full path to the MongoDB Server binaries. You should also add the path to `mongosh` if you have not already done so.

## Express JS

### Installing

Assuming you've already installed Node.js, create a directory to hold your application, and make that your working directory.

```
$ mkdir myapp  
$ cd myapp
```

Use the `npm init` command to create a `package.json` file for your application. For more information on how `package.json` works, see [Specifics of npm's package.json handling](#).

```
$ npm init
```

This command prompts you for a number of things, such as the name and version of your application. For now, you can simply hit RETURN to accept the defaults for most of them, with the following exception:

```
entry point: (index.js)
```

Enter `app.js`, or whatever you want the name of the main file to be. If you want it to be `index.js`, hit RETURN to accept the suggested default file name. Now install Express in the `myapp` directory and save it in the dependencies list. For example:

```
$ npm install express --save
```

To install Express temporarily and not add it to the dependencies list:

```
$ npm install express --no-save
```

## **Express application generator**

Use the application generator tool, `express-generator`, to quickly create an application skeleton.

You can run the application generator with the `npx` command (available in Node.js 8.2.0).

```
$ npx express-generator
```

For earlier Node versions, install the application generator as a global npm package and then launch it:

```
$ npm install -g express-generator
```

```
$ express
```

Display the command options with the `-h` option:



```
$ express -h
```

```
Usage: express [options] [dir]
```

```
Options:
```

```
-h, --help      output usage information
  --version     output the version number
-e, --ejs       add ejs engine support
  --hbs        add handlebars engine support
  --pug        add pug engine support
-H, --hogan     add hogan.js engine support
  --no-view    generate without view engine
-v, --view <engine> add view <engine> support
(ejs|hbs|hjs|jade|pug|twig|vash) (defaults to jade)
-c, --css <engine> add stylesheet <engine> support
(less|stylus|compass|sass) (defaults to plain css)
  --git        add .gitignore
-f, --force    force on non-empty directory
```

For example, the following creates an Express app named *myapp*. The app will be created in a folder named *myapp* in the current working directory and the view engine will be set to Pug:

```
$ express --view=pug myapp
```

```
create : myapp
```

```
create : myapp/package.json
```

```
create : myapp/app.js
create : myapp/public
create : myapp/public/javascripts
create : myapp/public/images
create : myapp/routes
create : myapp/routes/index.js
create : myapp/routes/users.js
create : myapp/public/stylesheets
create : myapp/public/stylesheets/style.css
create : myapp/views
create : myapp/views/index.pug
create : myapp/views/layout.pug
create : myapp/views/error.pug
create : myapp/bin
create : myapp/bin/www
```

Then install dependencies:

```
$ cd myapp
$ npm install
```

On MacOS or Linux, run the app with this command:

```
$ DEBUG=myapp:* npm start
```

On Windows Command Prompt, use this command:

```
> set DEBUG=myapp:* & npm start
```

On Windows PowerShell, use this command:

```
PS> $env:DEBUG='myapp:*'; npm start
```

Then load <http://localhost:3000/> in your browser to access the app.

The generated app has the following directory structure:

```
.
├── app.js
├── bin
│   └── www
├── package.json
├── public
│   ├── images
│   ├── javascripts
│   └── stylesheets
│       └── style.css
├── routes
│   ├── index.js
│   └── users.js
└── views
    ├── error.pug
    ├── index.pug
    └── layout.pug
```

7 directories, 9 files

## React JS

### Add React to a Website

Use as little or as much React as you need.

React has been designed from the start for gradual adoption, and you can use as little or as much React as you need. Perhaps you only want to add some “sprinkles of interactivity” to an existing page. React components are a great way to do that.

The majority of websites aren't, and don't need to be, single-page apps.

With a few lines of code and no build tooling, try React in a small part of your website. You can then either gradually expand its presence, or keep it contained to a few dynamic widgets.

- [Add React in One Minute](#)
- [Optional: Try React with JSX](#) (no bundler necessary!)

### Add React in One Minute

In this section, we will show how to add a React component to an existing HTML page. You can follow along with your own website, or create an empty HTML file to practice.

There will be no complicated tools or install requirements — to complete this section, you only need an internet connection, and a minute of your time.

Optional: Download the full example (2KB zipped)

#### Step 1: Add a DOM Container to the HTML

First, open the HTML page you want to edit. Add an empty `<div>` tag to mark the spot where you want to display something with React. For example:

```
<!-- ... existing HTML ... -->
```

```
<div id="like_button_container"></div>
<!-- ... existing HTML ... -->
```

We gave this `<div>` a unique id HTML attribute. This will allow us to find it from the JavaScript code later and display a React component inside of it.

### Tip

You can place a “container” `<div>` like this anywhere inside the `<body>` tag. You may have as many independent DOM containers on one page as you need. They are usually empty — React will replace any existing content inside DOM containers.

## Step 2: Add the Script Tags

Next, add three `<script>` tags to the HTML page right before the closing `</body>` tag:

```
<!-- ... other HTML ... -->

<!-- Load React. -->
<!-- Note: when deploying, replace "development.js" with
"production.min.js". -->
<script src="https://unpkg.com/react@17/umd/react.development.js"
crossorigin></script> <script src="https://unpkg.com/react-
dom@17/umd/react-dom.development.js" crossorigin></script>
<!-- Load our React component. -->
<script src="like_button.js"></script>
</body>
```

The first two tags load React. The third one will load your component code.

## Step 3: Create a React Component

Create a file called `like_button.js` next to your HTML page.

Open this starter code and paste it into the file you created.

### Tip

This code defines a React component called LikeButton. Don't worry if you don't understand it yet — we'll cover the building blocks of React later in our hands-on tutorial and main concepts guide. For now, let's just get it showing on the screen!

After the starter code, add two lines to the bottom of like\_button.js:

```
// ... the starter code you pasted ...  
  
const domContainer =  
document.querySelector('#like_button_container');ReactDOM.render(e(LikeB  
utton), domContainer);
```

These two lines of code find the <div> we added to our HTML in the first step, and then display our “Like” button React component inside of it.

## Quickly Try JSX

The quickest way to try JSX in your project is to add this <script> tag to your page:

```
<script src="https://unpkg.com/babel-standalone@6/babel.min.js"></script>
```

Now you can use JSX in any <script> tag by adding type="text/babel" attribute to it. Here is an example HTML file with JSX that you can download and play with.

This approach is fine for learning and creating simple demos. However, it makes your website slow and isn't suitable for production. When you're ready to move forward, remove this new <script> tag and the type="text/babel" attributes you've added. Instead, in the next section you will set up a JSX preprocessor to convert all your <script> tags automatically.

## Add JSX to a Project

Adding JSX to a project doesn't require complicated tools like a bundler or a development server. Essentially, adding JSX is a lot like adding a CSS preprocessor. The only requirement is to have Node.js installed on your computer.

Go to your project folder in the terminal, and paste these two commands:

**Step 1:** Run `npm init -y` (if it fails, here's a fix)

**Step 2:** Run `npm install babel-cli@6 babel-preset-react-app@3`

### Tip

We're using npm here only to install the JSX preprocessor; you won't need it for anything else. Both React and the application code can stay as `<script>` tags with no changes.

Congratulations! You just added a production-ready JSX setup to your project.

## Run JSX Preprocessor

Create a folder called `src` and run this terminal command:

```
npx babel --watch src --out-dir . --presets react-app/prod
```

### Note

`npx` is not a typo — it's a package runner tool that comes with npm 5.2+.

If you see an error message saying "You have mistakenly installed the babel package", you might have missed the previous step. Perform it in the same folder, and then try again.

Don't wait for it to finish — this command starts an automated watcher for JSX.

If you now create a file called `src/like_button.js` with this JSX starter code, the watcher will create a preprocessed `like_button.js` with the plain JavaScript

code suitable for the browser. When you edit the source file with JSX, the transform will re-run automatically.

As a bonus, this also lets you use modern JavaScript syntax features like classes without worrying about breaking older browsers. The tool we just used is called Babel, and you can learn more about it from its documentation. If you notice that you're getting comfortable with build tools and want them to do more for you, the next section describes some of the most popular and approachable toolchains.

## **React Top-Level API**

React is the entry point to the React library. If you load React from a `<script>` tag, these top-level APIs are available on the React global. If you use ES6 with npm, you can write `import React from 'react'`. If you use ES5 with npm, you can write `var React = require('react')`.

## **Overview**

### **Components**

React components let you split the UI into independent, reusable pieces, and think about each piece in isolation. React components can be defined by subclassing `React.Component` or `React.PureComponent`.

`React.Component`

`React.PureComponent`

If you don't use ES6 classes, you may use the `create-react-class` module instead. See [Using React without ES6](#) for more information.

React components can also be defined as functions which can be wrapped:

`React.memo`



## **Creating React Elements**

We recommend using JSX to describe what your UI should look like. Each JSX element is just syntactic sugar for calling `React.createElement()`. You will not typically invoke the following methods directly if you are using JSX.

`createElement()`

`createFactory()`

See [Using React without JSX](#) for more information.

## **Transforming Elements**

React provides several APIs for manipulating elements:

`cloneElement()`

`isValidElement()`

`React.Children`

## **Fragments**

React also provides a component for rendering multiple elements without a wrapper.

`React.Fragment`

## **Refs**

`React.createRef`

`React.forwardRef`

## **Suspense**

Suspense lets components “wait” for something before rendering. Today, Suspense only supports one use case: loading components dynamically

with `React.lazy`. In the future, it will support other use cases like data fetching.

`React.lazy`

`React.Suspense`

## Hooks

*Hooks* are a new addition in React 16.8. They let you use state and other React features without writing a class. Hooks have a dedicated docs section and a separate API reference:

- **Basic Hooks**
  - `useState`
  - `useEffect`
  - `useContext`
  
- **Additional Hooks**
  - `useReducer`
  - `useCallback`
  - `useMemo`
  - `useRef`
  - `useImperativeHandle`
  - `useLayoutEffect`
  - `useDebugValue`

## Reference

### **React.Component**

`React.Component` is the base class for React components when they are defined using ES6 classes:

```
class Greeting extends React.Component {
```

```
render() {  
  return <h1>Hello, {this.props.name}</h1>;  
}  
}
```

See the [React.Component API Reference](#) for a list of methods and properties related to the base `React.Component` class.

## **React.PureComponent**

`React.PureComponent` is similar to `React.Component`. The difference between them is that `React.Component` doesn't implement `shouldComponentUpdate()`, but `React.PureComponent` implements it with a shallow prop and state comparison.

If your React component's `render()` function renders the same result given the same props and state, you can use `React.PureComponent` for a performance boost in some cases.

### **Note**

`React.PureComponent`'s `shouldComponentUpdate()` only shallowly compares the objects. If these contain complex data structures, it may produce false-negatives for deeper differences. Only extend `PureComponent` when you expect to have simple props and state, or use `forceUpdate()` when you know deep data structures have changed. Or, consider using immutable objects to facilitate fast comparisons of nested data.

Furthermore, `React.PureComponent`'s `shouldComponentUpdate()` skips prop updates for the whole component subtree. Make sure all the children components are also “pure”.

## **React.memo**

```
const MyComponent = React.memo(function MyComponent(props) {
```

```
/* render using props */  
});
```

React.memo is a higher order component.

If your component renders the same result given the same props, you can wrap it in a call to React.memo for a performance boost in some cases by memoizing the result. This means that React will skip rendering the component, and reuse the last rendered result.

React.memo only checks for prop changes. If your function component wrapped in React.memo has a useState, useReducer or useContext Hook in its implementation, it will still rerender when state or context change.

By default it will only shallowly compare complex objects in the props object. If you want control over the comparison, you can also provide a custom comparison function as the second argument.

```
function MyComponent(props) {  
  /* render using props */  
}  
function areEqual(prevProps, nextProps) {  
  /*  
   return true if passing nextProps to render would return  
   the same result as passing prevProps to render,  
   otherwise return false  
  */  
}  
export default React.memo(MyComponent, areEqual);
```

This method only exists as a performance optimization. Do not rely on it to “prevent” a render, as this can lead to bugs.

Note

Unlike the `shouldComponentUpdate()` method on class components, the `areEqual` function returns true if the props are equal and false if the props are not equal. This is the inverse from `shouldComponentUpdate`.

### **createElement()**

```
React.createElement(  
  type,  
  [props],  
  [...children]  
)
```

Create and return a new React element of the given type. The type argument can be either a tag name string (such as 'div' or 'span'), a React component type (a class or a function), or a React fragment type.

Code written with JSX will be converted to use `React.createElement()`. You will not typically invoke `React.createElement()` directly if you are using JSX. See [React Without JSX](#) to learn more.

---

### **cloneElement()**

```
React.cloneElement(  
  element,  
  [config],  
  [...children]  
)
```

Clone and return a new React element using `element` as the starting point. `config` should contain all new props, key, or ref. The resulting element will have the original element's props with the new props merged in

shallowly. New children will replace existing children. key and ref from the original element will be preserved if no key and ref present in the config.

React.cloneElement() is almost equivalent to:

```
<element.type {...element.props} {...props}>{children}</element.type>
```

However, it also preserves refs. This means that if you get a child with a ref on it, you won't accidentally steal it from your ancestor. You will get the same ref attached to your new element. The new ref or key will replace old ones if present.

This API was introduced as a replacement of the deprecated React.addons.cloneWithProps().

### **createFactory()**

```
React.createFactory(type)
```

Return a function that produces React elements of a given type.

Like React.createElement(), the type argument can be either a tag name string (such as 'div' or 'span'), a React component type (a class or a function), or a React fragment type.

This helper is considered legacy, and we encourage you to either use JSX or use React.createElement() directly instead.

You will not typically invoke React.createFactory() directly if you are using JSX. See React Without JSX to learn more.

---

### **isValidElement()**

```
React.isValidElement(object)
```

Verifies the object is a React element. Returns true or false.

### **React.Children**

React.Children provides utilities for dealing with the `this.props.children` opaque data structure.

React.Children.map

```
React.Children.map(children, function[(thisArg)])
```

Invokes a function on every immediate child contained within `children` with `this` set to `thisArg`. If `children` is an array it will be traversed and the function will be called for each child in the array. If `children` is null or undefined, this method will return null or undefined rather than an array.

### Note

If `children` is a Fragment it will be treated as a single child and not traversed.

React.Children.forEach

```
React.Children.forEach(children, function[(thisArg)])
```

Like `React.Children.map()` but does not return an array.

React.Children.count

```
React.Children.count(children)
```

Returns the total number of components in `children`, equal to the number of times that a callback passed to `map` or `forEach` would be invoked.

React.Children.only

```
React.Children.only(children)
```

Verifies that `children` has only one child (a React element) and returns it.

Otherwise this method throws an error.

### Note:

`React.Children.only()` does not accept the return value of `React.Children.map()` because it is an array rather than a React element.

React.Children.toArray

## `React.Children.toArray(children)`

Returns the children opaque data structure as a flat array with keys assigned to each child. Useful if you want to manipulate collections of children in your render methods, especially if you want to reorder or slice `this.props.children` before passing it down.

## **React.Fragment**

The `React.Fragment` component lets you return multiple elements in a `render()` method without creating an additional DOM element:

```
render() {  
  return (  
    <React.Fragment>  
      Some text.  
      <h2>A heading</h2>  
    </React.Fragment>  
  );  
}
```

You can also use it with the shorthand `<></>` syntax. For more information, see [React v16.2.0: Improved Support for Fragments](#).

## **React.createRef**

`React.createRef` creates a ref that can be attached to React elements via the `ref` attribute.

```
class MyComponent extends React.Component {  
  constructor(props) {  
    super(props);  
  
    this.inputRef = React.createRef(); }  
}
```



```

render() {
  return <input type="text" ref={this.inputRef} />; }

componentDidMount() {
  this.inputRef.current.focus(); }
}

```

## React.forwardRef

React.forwardRef creates a React component that forwards the ref attribute it receives to another component below in the tree. This technique is not very common but is particularly useful in two scenarios:

Forwarding refs to DOM components

Forwarding refs in higher-order-components

React.forwardRef accepts a rendering function as an argument. React will call this function with props and ref as two arguments. This function should return a React node.

```

const FancyButton = React.forwardRef((props, ref) => (
  <button ref={ref}
  className="FancyButton"> {props.children}
  </button>
));

// You can now get a ref directly to the DOM button:
const ref = React.createRef();
<FancyButton ref={ref}>Click me!</FancyButton>;

```

In the above example, React passes a ref given to <FancyButton ref={ref}> element as a second argument to the rendering function inside

the `React.forwardRef` call. This rendering function passes the ref to the `<button ref={ref}>` element.

As a result, after React attaches the ref, `ref.current` will point directly to the `<button>` DOM element instance.

For more information, see forwarding refs.

## **React.lazy**

`React.lazy()` lets you define a component that is loaded dynamically. This helps reduce the bundle size to delay loading components that aren't used during the initial render.

You can learn how to use it from our code splitting documentation. You might also want to check out this article explaining how to use it in more detail.

```
// This component is loaded dynamically
const SomeComponent = React.lazy(() => import('./SomeComponent'));
```

Note that rendering lazy components requires that there's a `<React.Suspense>` component higher in the rendering tree. This is how you specify a loading indicator.

### **Note**

Using `React.lazy` with dynamic import requires Promises to be available in the JS environment. This requires a polyfill on IE11 and below.

## **React.Suspense**

`React.Suspense` lets you specify the loading indicator in case some components in the tree below it are not yet ready to render. Today, lazy loading components is the **only** use case supported by `<React.Suspense>`:

```
// This component is loaded dynamically
const OtherComponent = React.lazy(() => import('./OtherComponent'));
```

```
function MyComponent() {
  return (
    // Displays <Spinner> until OtherComponent loads
    <React.Suspense fallback={<Spinner />}>
      <div>
        <OtherComponent />
      </div>
    </React.Suspense>
  );
}
```

It is documented in our code splitting guide. Note that lazy components can be deep inside the Suspense tree — it doesn't have to wrap every one of them. The best practice is to place `<Suspense>` where you want to see a loading indicator, but to use `lazy()` wherever you want to do code splitting. While this is not supported today, in the future we plan to let Suspense handle more scenarios such as data fetching. You can read about this in our roadmap.

## Node.js

As an asynchronous event-driven JavaScript runtime, Node.js is designed to build scalable network applications. In the following "hello world" example, many connections can be handled concurrently. Upon each connection, the callback is fired, but if there is no work to be done, Node.js will sleep.

```
const http = require('http');

const hostname = '127.0.0.1';
```

```

const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});

```

This is in contrast to today's more common concurrency model, in which OS threads are employed. Thread-based networking is relatively inefficient and very difficult to use. Furthermore, users of Node.js are free from worries of dead-locking the process, since there are no locks. Almost no function in Node.js directly performs I/O, so the process never blocks except when the I/O is performed using synchronous methods of Node.js standard library. Because nothing blocks, scalable systems are very reasonable to develop in Node.js.

If some of this language is unfamiliar, there is a full article on [Blocking vs. Non-Blocking](#).

Node.js is similar in design to, and influenced by, systems like Ruby's Event Machine and Python's Twisted. Node.js takes the event model a bit further. It presents an event loop as a runtime construct instead of as a library. In other

systems, there is always a blocking call to start the event-loop. Typically, behavior is defined through callbacks at the beginning of a script, and at the end a server is started through a blocking call like `EventMachine::run()`. In Node.js, there is no such start-the-event-loop call. Node.js simply enters the event loop after executing the input script. Node.js exits the event loop when there are no more callbacks to perform. This behavior is like browser JavaScript — the event loop is hidden from the user.

HTTP is a first-class citizen in Node.js, designed with streaming and low latency in mind. This makes Node.js well suited for the foundation of a web library or framework.

Node.js being designed without threads doesn't mean you can't take advantage of multiple cores in your environment. Child processes can be spawned by using our `child_process.fork()` API, and are designed to be easy to communicate with. Built upon that same interface is the `cluster` module, which allows you to share sockets between processes to enable load balancing over your cores.

## HTTP#

To use the HTTP server and client one must `require('http')`.

The HTTP interfaces in Node.js are designed to support many features of the protocol which have been traditionally difficult to use. In particular, large, possibly chunk-encoded, messages. The interface is careful to never buffer entire requests or responses, so the user is able to stream data.

HTTP message headers are represented by an object like this:

```
{ 'content-length': '123',  
  'content-type': 'text/plain',  
  'connection': 'keep-alive',  
  'host': 'mysite.com',  
  'accept': '*/*' }
```

Keys are lowercased. Values are not modified.

In order to support the full spectrum of possible HTTP applications, the Node.js HTTP API is very low-level. It deals with stream handling and message parsing only. It parses a message into headers and body but it does not parse the actual headers or the body.

See `message.headers` for details on how duplicate headers are handled.

The raw headers as they were received are retained in the `rawHeaders` property, which is an array of `[key, value, key2, value2, ...]`.

For example, the previous message header object might have a `rawHeaders` list like the following:

```
[ 'ConTent-Length', '123456',  
  'content-LENGTH', '123',  
  'content-type', 'text/plain',  
  'CONNECTION', 'keep-alive',  
  'Host', 'mysite.com',  
  'accepT', '*/*' ]
```

## **Results and Discussions**

Taking up a new work in a comparatively new technology like MERN stack leaves us with new experiences and a huge amount of new knowledge. This project was a very challenging one for both of us, but with the help of various references, the continuous support of our guide and the internet, we were able to complete our visioned project in time.

There were quite many challenges in linking the different modules once they were ready. All the errors took us a lot of research through GitHub and rectifying them took time.

Hence, with a lot of hard work and constant support we were able to finish the project work. It may not be perfect and there may be a number of errors, but we put our best in the development of the project and will continuously work on its improvement in the future.

## Conclusion

Like any other stack, you can use the MERN stack to build any application you want. Although it is ideally used where the application is cloud-native, JSON-heavy, and need dynamic web interfaces like Todo apps, workflow management, interactive forums, and social products.

So, if you want to develop an app that is cost-effective, open-source, and offers better performance and UI rendering, you can always opt for the MERN stack. The project itself gave us a lot of challenges and we also got to learn about many new technologies and ways to implement ideas into real world application.

We have tried to keep the project as useful and as glitch free as possible, but nevertheless, there may be some nits and glitches. The lifecycle of a software is the improvement in its performance with continuous updates and this project of ours will also continue to do so.



## References

- [1] J. Kwon and S. Moon, "Work-in-progress: JSDelta: serializing modified javascript states for state sharing," 2017 International Conference on Embedded Software (EMSOFT), Seoul, 2017, pp.1-2.
- [2] J. Heo, S. Woo, H. Jang, K. Yang and J. W. Lee, "Improving JavaScript performance via efficient in-memory bytecode caching," 2016 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia), Seoul, 2016, pp.1-4.
- [3] H. Park, W. Jung and S. Moon, "Javascript ahead-of-time compilation for embedded web platform," 2015 13th IEEE Symposium on Embedded Systems For Real-time Multimedia (ESTIMedia), Amsterdam, 2015, pp.1-9.
- [4] G. Prabagaren, "Systematic approach for validating Java-MongoDB Schema," International Conference on Information Communication and Embedded Systems (ICICES2014), Chennai, 2014, pp.1-4.
- [5] Velliangiri, S., Karthikeyan, P., Xavier, V. A., & Baswaraj, D. (2021). Hybrid electro search with genetic algorithm for task scheduling in cloud computing. *Ain Shams Engineering Journal*, 12(1), 631-639.
- [6] J. Kumar and V. Garg, "Security analysis of unstructured data in NOSQL MongoDB database," 2017 International Conference on Computing and Communication Technologies for Smart Nation (IC3TSN), Gurgaon, 2017, pp. 300-305.