

**SIX WEEKS SUMMER TRAINING REPORT**

On

**Full-Stack Development**

Submitted by

Your Name

Registration No. 1200000

Program Name: Bachelor of Technology

Under the Guidance of

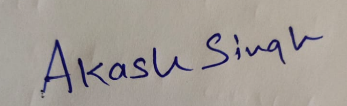
**Mr. Anurag Mishra**

School of Computer Science & Engineering Lovely professional University, Phagwara

(May-July, 2022)

**DECLARATION**

I hereby declare that I have completed my six weeks summer training at Full-Stack Development, CipherSchools from 25th May 2022 to 10th July 2022 under the guidance of Mr. Anurag Mishra. I have declare that I have worked with full dedication during these six weeks of training and my learning outcomes fulfil the requirements of training for the award of degree of Bachelor of Technology, Lovely Professional university, Phagwara.



(Signature of student)

Name of Student: Your Name

Registration no: 1200000

Date: 24th Sept, 2022

#### Acknowledgement

It is with sense of gratitude; I acknowledge the efforts of entire hosts of well-wishers who have in some way or other contributed in their own special ways to the success and completion of the Summer Training.

Successfully completion of any type technology requires helps from a number of people. I have also taken help from different people for the preparation of the report. Now, there is little efforts to show my deep gratitude to those helpful people.

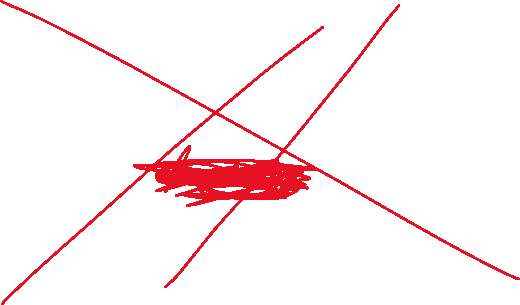
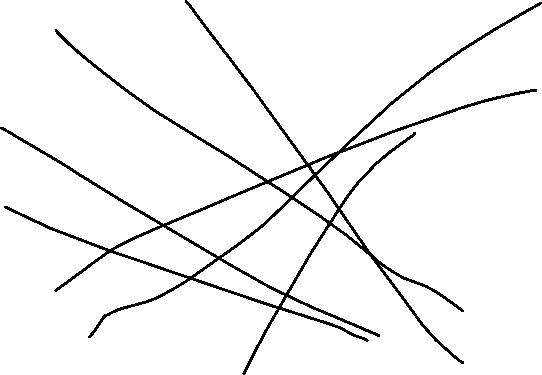
I would like to also thank my own college Lovely Professional University for offering such a course which not only improve my programming skill but also taught me other new technology.

Then I would like to thank my parents and friends who have helped me with their valuable suggestions and guidance for choosing this course.

Last but not least I would like to thank my all classmates who have helped me a lot.

**Training certificate from organization**





**Table Of Contents**

1. Introduction
2. Technology Learnt
3. Reason for choosing this technology.
4. Profile of the Problem
5. Existing System
6. Problem Analysis
   * Product definition
7. Software Requirement Analysis
8. Design
   * Tables and their relationships
   * Flowcharts/Pseudo code
9. Implementation
10. Leaning Outcome from training/technology learnt
11. Gantt chart
12. Project Legacy
    * Technical and Managerial learnt.
13. Bibliography

**Introduction**

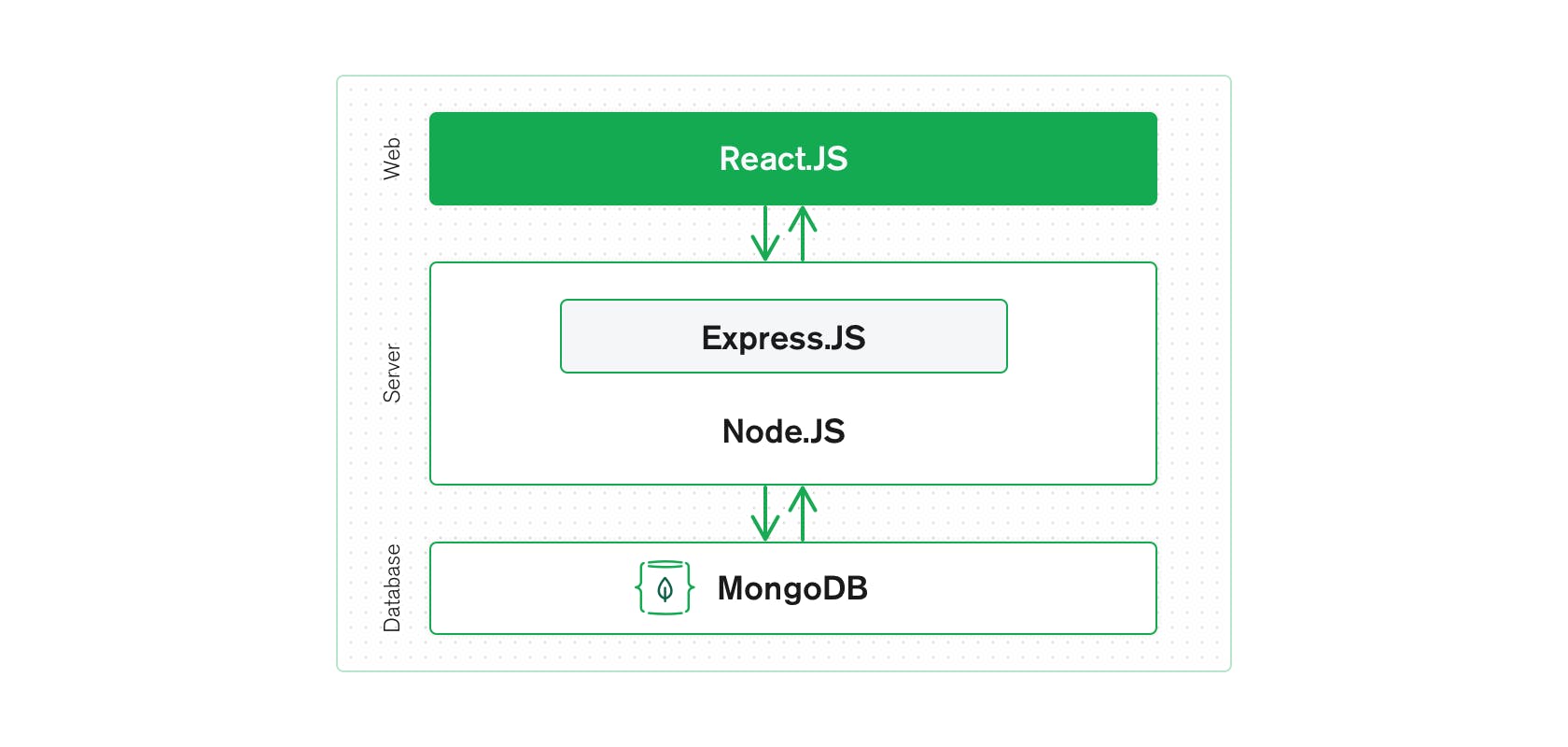
MERN stands for MongoDB, Express, React, Node, after the four key technologies that make up the stack.

* MongoDB — document database
* Express(.js) — Node.js web framework
* React(.js) — a client-side JavaScript framework
* Node(.js) — the premier JavaScript web server

Express and Node make up the middle (application) tier. Express.js is a server-side web framework, and Node.js is the popular and powerful JavaScript server platform. Regardless of which variant you choose, ME(RVA)N is the ideal approach to working with JavaScript and JSON, all the way through.

**How does the MERN stack work?**

The MERN architecture allows you to easily construct a three-tier architecture (front end, back end, database) entirely using JavaScript and JSON.



**React.js front end**

The top tier of the MERN stack is React.js, the declarative JavaScript framework for creating dynamic client-side applications in HTML. React lets you build up complex interfaces through simple components, connect them to data on your back-end server, and render them as HTML.

React’s strong suit is handling stateful, data-driven interfaces with minimal code and minimal pain, and it has all the bells and whistles you’d expect from a modern web framework: great support for forms, error handling, events, lists, and more.

**Express.js and Node.js server tier**

The next level down is the Express.js server-side framework, running inside a Node.js server. Express.js bills itself as a “fast, unopinionated, minimalist web framework for Node.js,” and that is indeed exactly what it is. Express.js has powerful models for URL routing (matching an incoming URL with a server function), and handling HTTP requests and responses.

By making XML HTTP Requests (XHRs) or GETs or POSTs from your React.js front end, you can connect to Express.js functions that power your application. Those functions, in turn, use MongoDB’s Node.js drivers, either via callbacks or using promises, to access and update data in your MongoDB database.

**MongoDB database tier**

If your application stores any data (user profiles, content, comments, uploads, events, etc.), then you’re going to want a database that’s just as easy to work with as React, Express, and Node.

That’s where MongoDB comes in: JSON documents created in your React.js front end can be sent to the Express.js server, where they can be processed and (assuming they’re valid) stored directly in MongoDB for later retrieval. Again, if you’re building in the cloud, you’ll want to look at Atlas. If you’re looking to set up your own MERN stack, read on!

**Technology Learnt**

* Getting familiar with dev environment: Linux, Understanding common Linux commands.
* Version Control System and GitHub, Common Git commands: add, push, pull, diff, clone, etc. Git branching.
* Understanding how website works – Website Components like Backend, frontend, databases. What are REST APIs. Browsers, HTTP vs HTTPS.
* CSS: layout, Flexbox and different flexbox tags.
* JavaScript and common JS concepts: JS engine, Event loop, Variable Hoisting, Event Bubbling and Propagation, event delegation, callbacks, Promises, Closures, Arrow Functions.
* ReactJs: Using create react app to create first app, npm, JSX, Components, Props and State.
* React Hooks: useState, useRef.
* React Context APIs: useContext and its implementations. Custom Hooks: useFetch.
* React Routers
* Connecting Nodejs server to Database: Schema and Model. Writing a small API to insert and fetch data from MongoDB.
* Understanding Error codes and API Responses, JSON.
* Middleware and Routes in Application Logic.
* Integrating frontend and backend using Axios.
* Authentication and Authorisation: concept of Tokens, Cookies, why they are needed and how they help in authentication and authorisation.
* Social Based Logic integration.
* Testing: Unit, E2E and other tests.

**Reason for choosing this technology**

The MERN stack. MongoDB was designed to store JSON data natively (it technically uses a binary version of JSON called BSON), and everything from its command line interface to its query language (MQL, or MongoDB Query Language) is built on JSON and JavaScript.

MongoDB works extremely well with Node.js, and makes storing, manipulating, and representing JSON data at every tier of your application incredibly easy. For cloud-native applications, MongoDB Atlas makes it even easier, by giving you an auto-scaling MongoDB cluster on the cloud provider of your choice, as easy as a few button clicks.

**Express.js** (running on Node.js) and React.js make the JavaScript/JSON application MERN full stack, well, full. Express.js is a server-side application framework that wraps HTTP requests and responses, and makes it easy to map URLs to server-side functions. React.js is a front-end JavaScript framework for building interactive user interfaces in HTML, and communicating with a remote server.

The combination means that JSON data flows naturally from front to back, making it fast to build on and reasonably simple to debug. Plus, you only have to know one programming language, and the JSON document structure, to understand the whole system!

MERN is the stack of choice for today’s web developers looking to move quickly, particularly for those with React.js experience.

**MERN use cases**

Like any web stack, you can build whatever you want in MERN — though it’s ideally suited for cases that are JSON-heavy, cloud-native, and that have dynamic web interfaces.

Examples include workflow management, news aggregation, to-do apps and calendars, and interactive forums/social products — and whatever else you can dream up!

## Profile of the Problem

1. There should be a list page to display the employee list
2. There should be a button to add a new employee
3. On click of this button, a form will open with all the fields mentioned in the employee The user should be able to fill the form and create a new employee.
4. The last column of the employee table will have two buttons, Edit and Activate / Deactivate
5. On click of Edit button, a form will open with auto populated employee data. The user should be able to edit details and save employees.
6. On click of the deactivate button, the user should be able to deactivate the employee.
7. On click of the activate button, the user should be able to activate the employee.

REST APIs

* Create Employee

Method: POST

Path: /api/user

Payload: All the fields mentioned in schema

* Update Employee

Method: PUT

Path: /api/user/:id

Payload: All the fields mentioned in schema-

* Get list of Employees

Method: GET

Path: /api/user

* Deactivate / Activate Employee

Method: PUT Path: /api/user/:id/:action

Note: If the action is ‘deactivate’, set DeletedAt key to current date.

If action is ‘activate’, set DeletedAt to null.

Do not remove the employee from the database.

### Problem Analysis

1. **Product definition**

Create Employee Management System MERN application.

### Software Requirement Analysis

1. Code Editor IDE
2. MongoDB
3. NodeJs
4. RectJs
5. PostMan
6. Operating System

**Design**

* **Flowcharts/Pseudo code**

**Front End(Using React.js)**

1. There should be a list page to display the employee list with following fields:
   1. First Name
   2. Last Name
   3. Email
   4. Mobile
2. There should be a button to add a new employee
3. On click of this button, a form will open with all the fields mentioned in the employee The user should be able to fill the form and create a new employee.
4. The last column of the employee table will have two buttons, Edit and Activate / Deactivate
5. On click of Edit button, a form will open with auto populated employee data. The user should be able to edit details and save employees
6. On click of the deactivate button, the user should be able to deactivate the employee.
7. On click of the activate button, the user should be able to activate the employee.

**Create Employee schema with the following fields:**

FirstName (String)

LastName (String)

Email (String)

Mobile (String)

Designation (String)

ReportingManager (String)

Salary (Number)

EmployeeCode (Number)

Location (String)

State (String)

Country (String)

Department (String)

**Write the following REST APIs for the following schema**

* Create Employee

Method: POST

Path: /api/user

Payload: All the fields mentioned in schema

* Update Employee

Method: PUT

Path: /api/user/:id

Payload: All the fields mentioned in schema-

* Get list of Employees

Method: GET

Path: /api/user

* Deactivate / Activate Employee

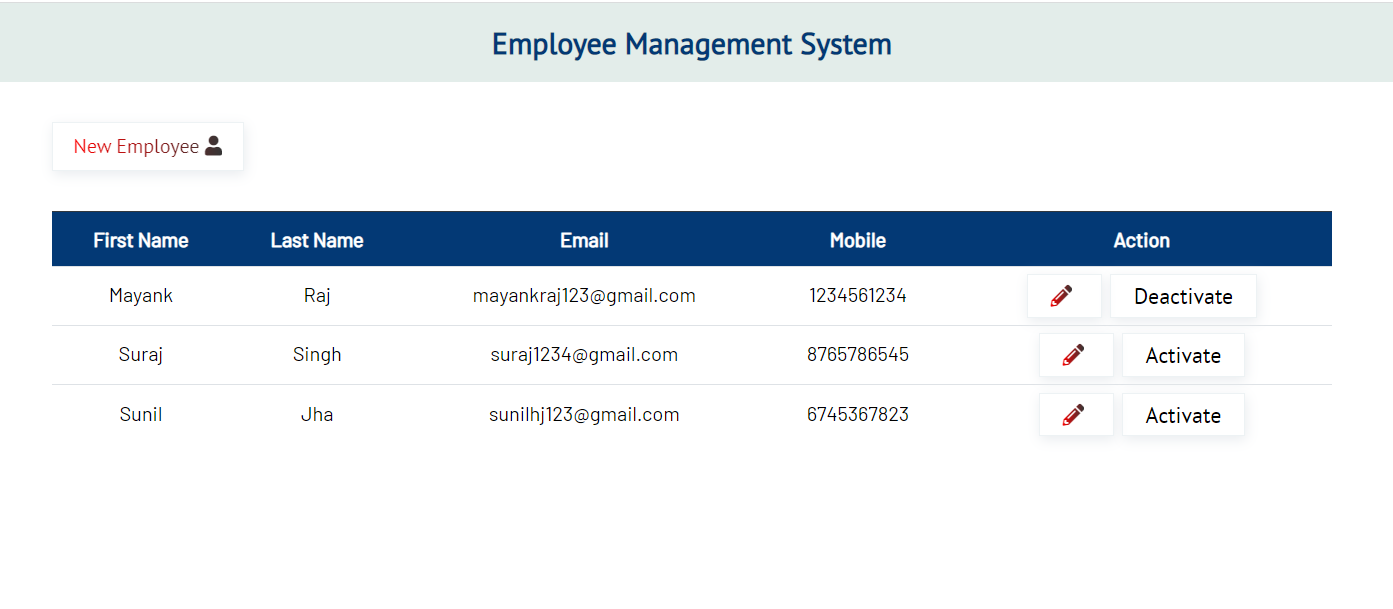
Method: PUT Path: /api/user/:id/:action

Note: If the action is ‘deactivate’, set DeletedAt key to current date.

If action is ‘activate’, set DeletedAt to null.

Do not remove the employee from the database.

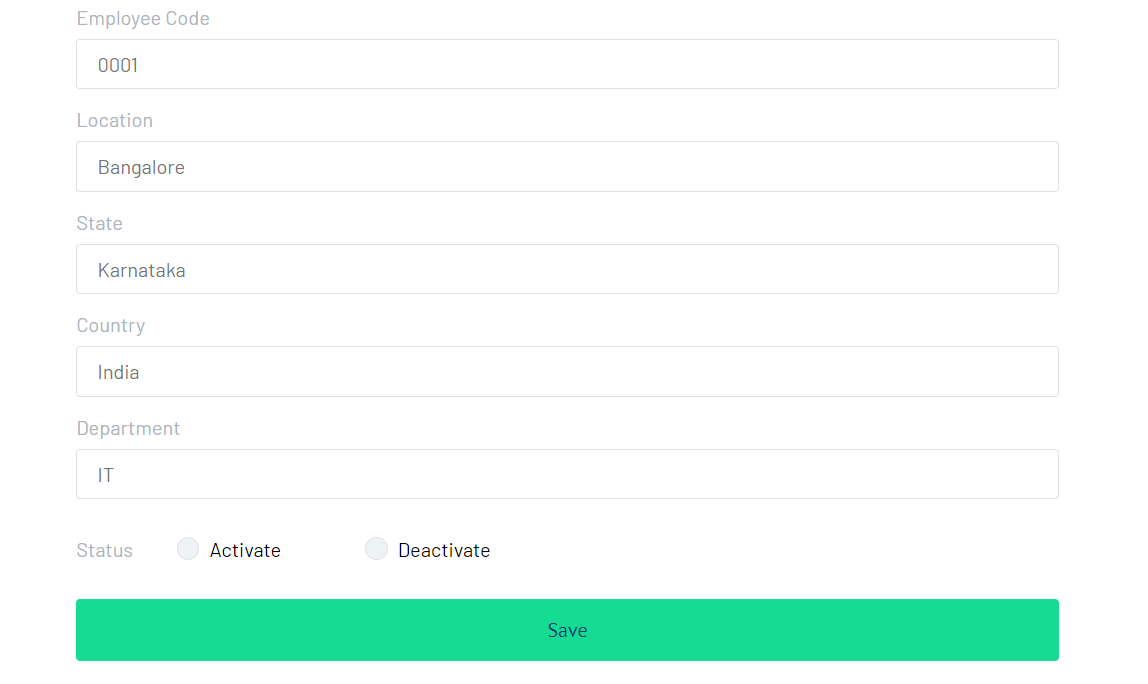
**Implementation**



##### Add Employee

Graphical user interface, application

Description automatically generated



##### Edit Employee Details

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, application, Teams

Description automatically generated

**Learning Outcomes**

* ReactJS’s component-based approach has reusability of codes, making app development faster and easier.
* The MERN stack also supports the Model View Controller architecture, allowing for a smooth and seamless development process.
* ReactJS’s implication becomes a fantastic framework for running the same code on the browser and the server.
* The MERN stack comes with a pre-built set of testing tools for app development.
* It has a thriving, active community and is completely open-source.
* This Stack is especially self-contained.
* Throughout the development cycle, it assumes all development-related responsibilities (i.e., from front-end development to back-end development)

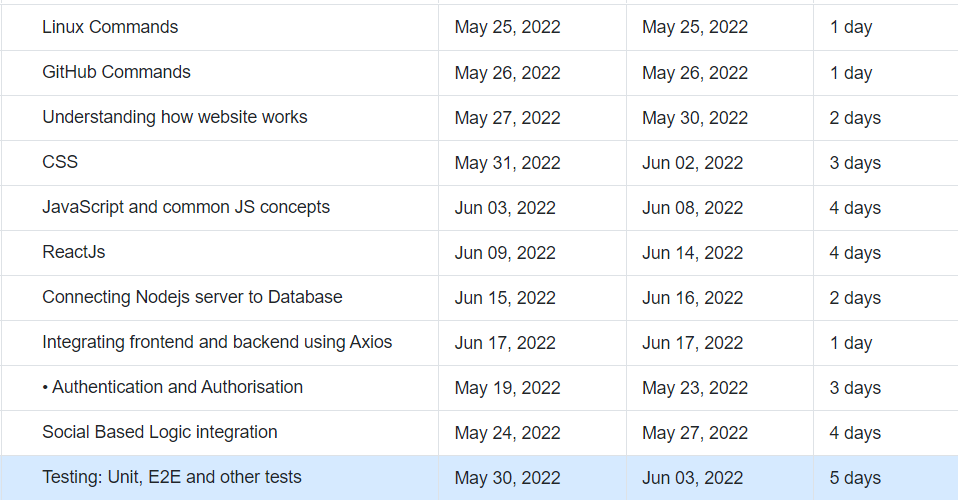
Timeline

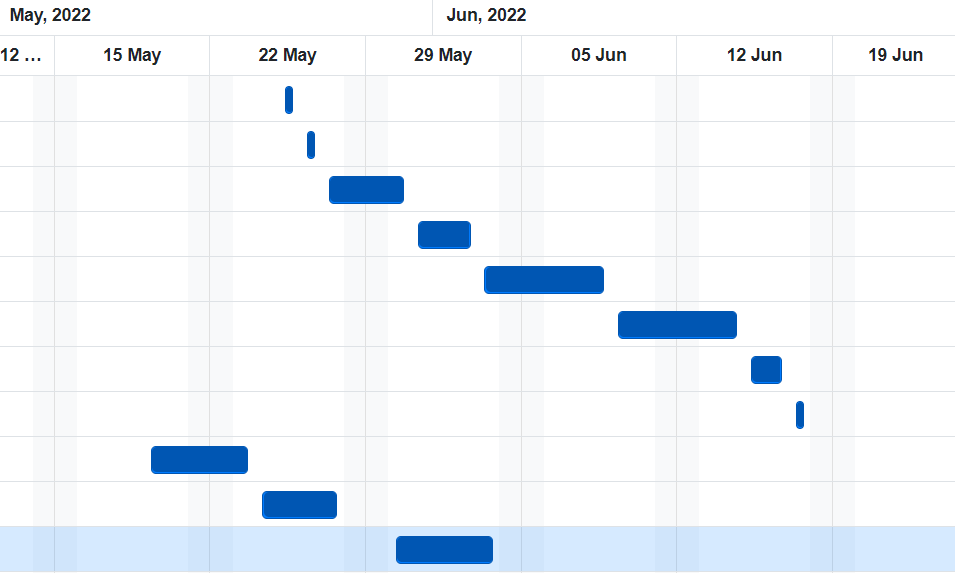
Description automatically generated

MERN Stack makes switching between front-end and back-end development much easier depending on the project requirements. MERN Stack is in charge of the venture’s planning infrastructure and works on a specific level during the development. It also saves money by hiring a full-stack developer rather than dedicated front-end and back-end developers.

* It is simple to improve with cutting-edge technology and equipment, and it also provides flexibility from one assignment to the next as a permission requirement.
* A start-up’s need for intelligence to communicate its ideas to the outside world is constant. Because of its JavaScript connection, MERN Stack is popular among developers.
* MERN Stack provides a complete development environment. End-to-end development is possible with this capability. Start-ups do not need to be concerned about other technologies or equipment because they can get everything, they need from the Stack itself.
* Holding a fair MERN Stack degree is regarded as one of the most valuable assets on any candidate’s resume, and it also aids the candidate in settling into large organizations.

**Gantt Chart**





**Bibliography**

* Google
* <https://nodejs.org/en/docs/>
* <https://www.npmjs.com/package/mongoose?activeTab=readme>
* YouTube