

# PROJECT REPORT

## On

# Learn2Earn

Computer Science & Engineering

**SUBMITTED BY**

Nitish Kumar  
University RollNo.:20012004501

**GUIDED BY**

Ms.Shilpi Gupta  
(Project Guide & HOD )

**anangpuria**

**DEPARTMENT OF COMPUTER SCIENCE AND  
ENGINEERING**

**B.S Anangpuria Institute of Technology &Management Alampur,  
Faridabad**

## **DECLARATION**

I hereby declare that the project entitled “**Learn2Earn**” submitted for the partial B.Tech. (CSE) degree is my original work and the project has not formed the basis for the award of any other degree, diploma, fellowship or any other similar titles.

Place: Faridabad

Date:

**Signature**

# **INDEX:-**

- 1. Abstract.**
- 2. Introduction.**
- 3. Proposed Methodology**
- 4. Flowchart**
- 5. Detailed Information**
- 6. Results**
- 7. Future Scope**
- 8. References**

## **ABSTRACT**

- Learn2Earn is premium course selling platform where student can register themselves in order to upskill.
- This application is solution to for all course selling businesses in order to run their venture.
- It provides subscription based model for courses. It simply means that any learner with premium membership credential can access the course.
- It is MERN based web application which is fast in terms of response time.
- Main aim of this system is to provide facility to students that they learn couese easily without any problems that they face while going in the institute.

# INTRODUCTION

Learn2Earn is a course selling web app built on Javascript Framework. Frontend is developed using React.js and its backend is developed in Express.js. MongoDB is used for database part. The main aim of this web app is to provide a solution to course selling businesses for their scalability. This website response time is faster than the others website.

Learn2Earn is a Full Stack Web Application which are oftenly referred as Frontend + Backend but there's more to it than just Frontend and Backend. Terms such as security, how much lightweight, fast the application is, even it's live on Internet or not means the servers are running your code 24x7 basically, people can access your website any time any where, A separated Environment for Development and Production. All of this comes under the Full Stack umbrella.

This project is made somewhat like an online edtech platform like Unacademy. This Project mimics some of the functionality that we see in online edtech website like a user can Signin/Signup, can see some course, write review, buy subscription.

## **Tools Used**

### **Visual Studio Code**

This is IDE which allow us to code efficiently.It provides access to terminal which make development friendly environment . It has large set of useful extension which reduces the pain of developer.

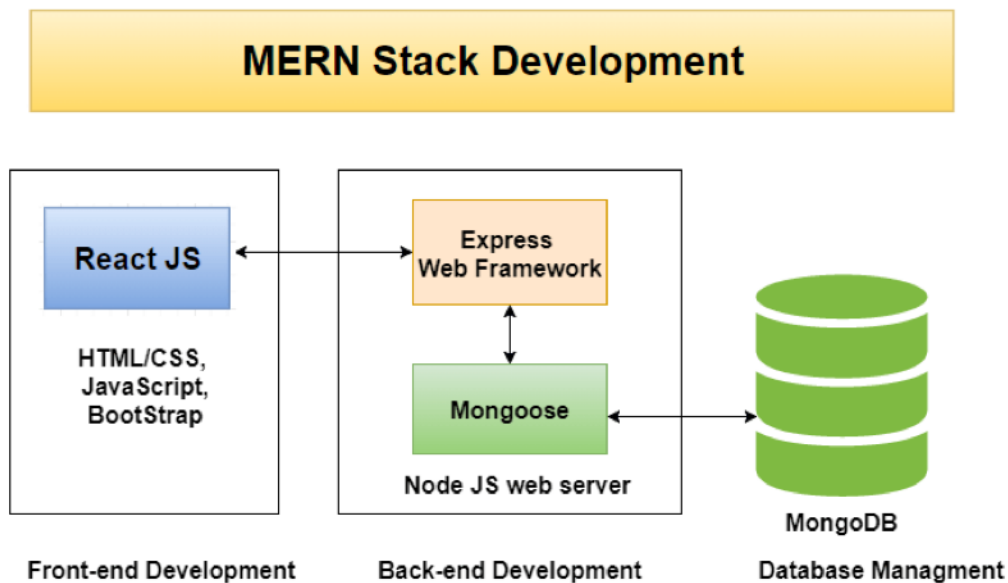
### **MongoDB Compass**

This is an interactive tool for querying, optimizing, and analyzing your **MongoDB** data.

### **POSTMAN**

Postman is an API platform for developers to design, build, test and iterate their APIs.

## Proposed Methodology



## TechStack/Libraries/Frameworks

### Node.js

Node.js is an open-source server environment. Node.js is cross-platform and runs on Windows, Linux, Unix, and macOS. Node.js is a back-end JavaScript runtime environment. Node.js runs on the V8 JavaScript Engine and executes JavaScript code outside a web browser.

### **Frontend**

As a core technology frontend part contains :

- **HTML**

The HyperText Markup Language or HTML is the standard markup language for documents designed to be displayed in a web browser.

- **CSS**

Cascading Style Sheets is a style sheet language used for describing the presentation of a document written in a markup language such as HTML or XML.

- **Javascript**

JavaScript, often abbreviated as JS, is a programming language that is one of the core technologies of the World Wide Web, alongside HTML and CSS. As of 2022, 98% of websites use JavaScript on the client side for webpage behavior, often incorporating third-party libraries.

- **React.js**

React is a free and open-source front-end JavaScript library for building user interfaces based on UI components. It is maintained by Meta and a community of individual developers and companies.

## **Backend**

For server development I have used Express.js and MongoDB for database service.

- **Express.js**

Express.js, or simply Express, is a back end web application framework for building RESTful APIs with Node.js, released as

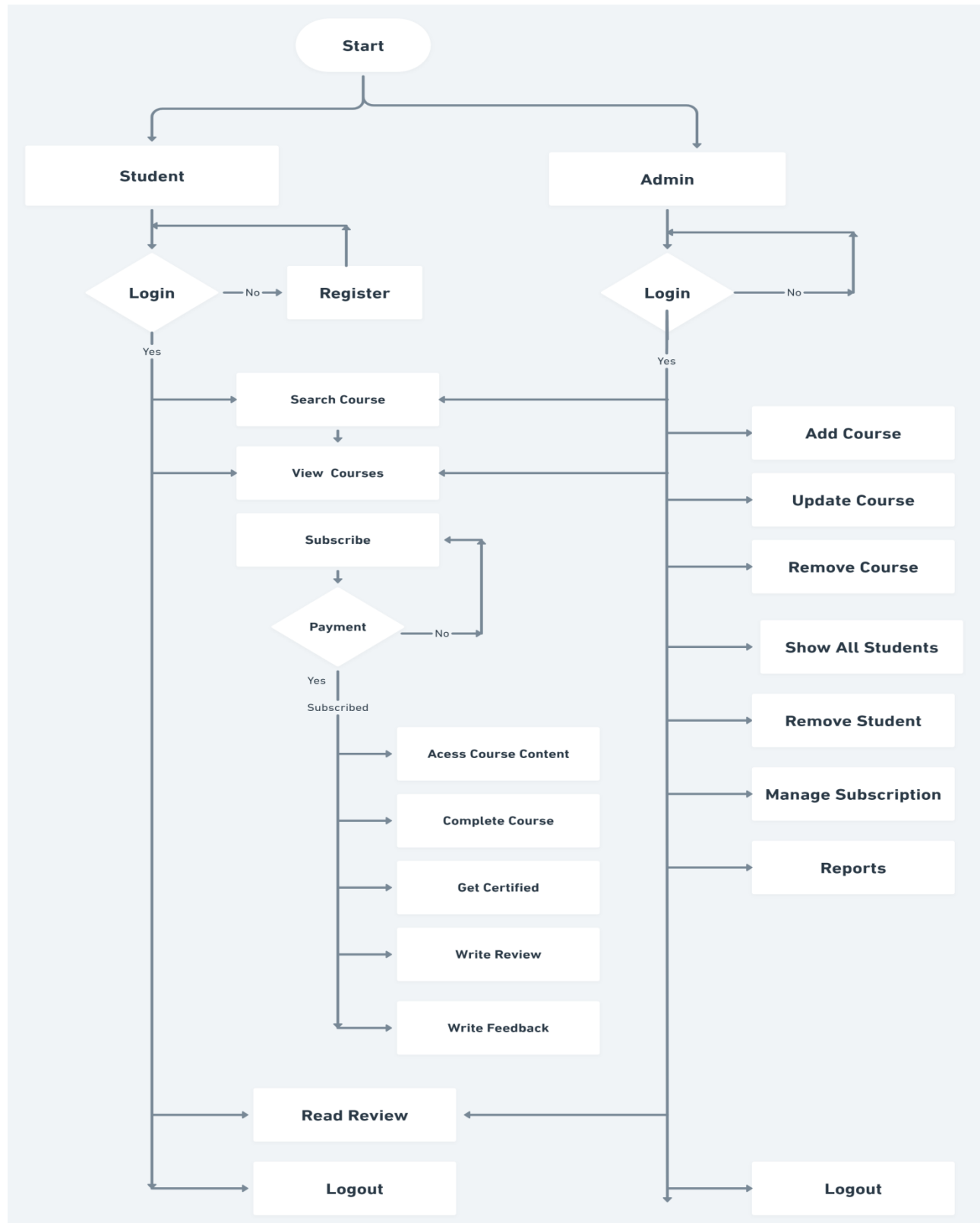


free and open-source software under the MIT License. It is designed for building web applications and APIs. It has been called the de facto standard server framework for Node.js.

- **Mongodb**

MongoDB is a source-available cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas. MongoDB is developed by MongoDB Inc. and licensed under the Server Side Public License which is deemed non-free by several distributions.

# Flowchart



# Detailed Information

## Course Schema

```
const schema = new mongoose.Schema({
  title: {
    type: String,
    required: [true, "Please enter course title"],
    minLength: [4, "Title must be at least 4 characters"],
    maxLength: [80, "Title can't exceed 80 characters"],
  },
  description: {
    type: String,
    required: [true, "Please enter course title"],
    minLength: [20, "Title must be at least 20 characters"],
  },
  lectures: [...],
  poster: {
    public_id: {
      type: String,
      required: true,
    },
    url: {
      type: String,
      required: true,
    },
  },
  views: {
    type: Number,
    default: 0,
  },
  numOfVideos: {
    type: Number,
    default: 0,
  },
  category: {
    type: String,
    required: true,
  },
  createdBy: {
    type: String,
    required: [true, "Enter Course Creator Name"],
  },
  createdAt: {
    type: Date,
    default: Date.now,
  },
});
```

# User Schema

```
const schema = new mongoose.Schema({
  name: {
    type: String,
    required: [true, "Please enter your name"],
  },
  email: {
    type: String,
    required: [true, "Please enter your email"],
    unique: true,
    validate: validator.isEmail,
  },

  password: {
    type: String,
    required: [true, "Please enter your password"],
    minLength: [6, "Password must be at least 6 characters"],
    select: false,
  },
  role: {
    type: String,
    enum: ["admin", "user"],
    default: "user",
  },

  subscription: {
    id: String,
    status: String,
  },

  > avatar: { ...
  },

  > playlist: [ ...
  ],

  createdAt: {
    type: Date,
    default: Date.now,
  },

  resetPasswordToken: String,
  resetPasswordExpire: String,
});
```

**Course Controller** : Contain all operation to of course entity.

```
export const getAllCourses = catchAsyncError(async (req, res, next) =>
{
  const keyword = req.query.keyword || "";
  const category = req.query.category || "";

  const courses = await Course.find({
    title: {
      $regex: keyword,
      $options: "i",
    },
    category: {
      $regex: category,
      $options: "i",
    },
  }).select("-lectures");
  res.status(200).json({
    success: true,
    courses,
  });
});

export const createCourse = catchAsyncError(async (req, res, next) =>
{
  const { title, description, category, createdBy } = req.body;

  if (!title || !description || !category || !createdBy)
    return next(new ErrorHandler("Please add all fields", 400));

  const file = req.file;

  const fileUri = getDataUri(file);

  const mycloud = await
cloudinary.v2.uploader.upload(fileUri.content);

  await Course.create({
    title,
    description,
    category,
```

```
        createdBy,  
        poster: {  
          public_id: mycloud.public_id,  
          url: mycloud.secure_url,  
        },  
      });  
  
    res.status(201).json({  
      success: true,  
      message: "Course Created Successfully. You can add lectures now.",  
    });  
  });  
  
export const getCourseLectures = catchAsyncError(async (req, res,  
next) => {  
  const course = await Course.findById(req.params.id);  
  
  if (!course) return next(new ErrorHandler("Course not found", 404));  
  
  course.views += 1;  
  
  await course.save();  
  
  res.status(200).json({  
    success: true,  
    lectures: course.lectures,  
  });  
});  
  
// Max video size 100mb  
export const addLecture = catchAsyncError(async (req, res, next) => {  
  const { id } = req.params;  
  const { title, description } = req.body;  
  
  const course = await Course.findById(id);  
  
  if (!course) return next(new ErrorHandler("Course not found", 404));  
  
  const file = req.file;  
  const fileUri = getDataUri(file);
```

```

    const mycloud = await cloudinary.v2.uploader.upload(fileUri.content,
    {
        resource_type: "video",
    });

    course.lectures.push({
        title,
        description,
        video: {
            public_id: mycloud.public_id,
            url: mycloud.secure_url,
        },
    });

    course.numOfVideos = course.lectures.length;

    await course.save();

    res.status(200).json({
        success: true,
        message: "Lecture added in Course",
    });
});

export const deleteCourse = catchAsyncError(async (req, res, next) =>
{
    const { id } = req.params;

    const course = await Course.findById(id);

    if (!course) return next(new ErrorHandler("Course not found", 404));

    await cloudinary.v2.uploader.destroy(course.poster.public_id);

    for (let i = 0; i < course.lectures.length; i++) {
        const singleLecture = course.lectures[i];
        await
cloudinary.v2.uploader.destroy(singleLecture.video.public_id, {
            resource_type: "video",
        });
    }
}

```

```
    await course.remove();

    res.status(200).json({
      success: true,
      message: "Course Deleted Successfully",
    });
  });
});

export const deleteLecture = catchAsyncError(async (req, res, next) => {
  const { courseId, lectureId } = req.query;

  const course = await Course.findById(courseId);
  if (!course) return next(new ErrorHandler("Course not found", 404));

  const lecture = course.lectures.find((item) => {
    if (item._id.toString() === lectureId.toString()) return item;
  });
  await cloudinary.v2.uploader.destroy(lecture.video.public_id, {
    resource_type: "video",
  });

  course.lectures = course.lectures.filter((item) => {
    if (item._id.toString() !== lectureId.toString()) return item;
  });

  course.numOfVideos = course.lectures.length;

  await course.save();

  res.status(200).json({
    success: true,
    message: "Lecture Deleted Successfully",
  });
});

Course.watch().on("change", async () => {
  const stats = await Stats.find({}).sort({ createdAt: "desc"
}).limit(1);
```



```

const courses = await Course.find({});

let totalViews = 0;

for (let i = 0; i < courses.length; i++) {
  totalViews += courses[i].views;
}
stats[0].views = totalViews;
stats[0].createdAt = new Date(Date.now());

await stats[0].save();
});

```

**User Controller :** Contain all operation to of user entity.

```

const schema = new mongoose.Schema({
  name: {
    type: String,
    required: [true, "Please enter your name"],
  },
  email: {
    type: String,
    required: [true, "Please enter your email"],
    unique: true,
    validate: validator.isEmail,
  },
  password: {
    type: String,
    required: [true, "Please enter your password"],
    minLength: [6, "Password must be at least 6 characters"],
    select: false,
  },
  role: {
    type: String,
    enum: ["admin", "user"],
    default: "user",
  },
});

```

```
subscription: {
  id: String,
  status: String,
},

avatar: {
  public_id: {
    type: String,
    required: true,
  },
  url: {
    type: String,
    required: true,
  },
},

playlist: [
  {
    course: {
      type: mongoose.Schema.Types.ObjectId,
      ref: "Course",
    },
    poster: String,
  },
],

createdAt: {
  type: Date,
  default: Date.now,
},

resetPasswordToken: String,
resetPasswordExpire: String,
});

schema.pre("save", async function (next) {
  if (!this.isModified("password")) return next();
  this.password = await bcrypt.hash(this.password, 10);
  next();
});
```

```
schema.methods.getJWTToken = function () {  
  return jwt.sign({ _id: this._id }, process.env.JWT_SECRET, {  
    expiresIn: "15d",  
  });  
};  
  
schema.methods.comparePassword = async function (password) {  
  return await bcrypt.compare(password, this.password);  
};  
  
schema.methods.getResetToken = function () {  
  const resetToken = crypto.randomBytes(20).toString("hex");  
  
  this.resetPasswordToken = crypto  
    .createHash("sha256")  
    .update(resetToken)  
    .digest("hex");  
  
  this.resetPasswordExpire = Date.now() + 15 * 60 * 1000;  
  
  return resetToken;  
};
```

# Results

## (1)Landing Page



### LEARN FROM THE EXPERTS

Find Valuable Content At Reasonable Price

Explore Now



Learn2Earn

Home

Browse All Courses

Request a Course

Contact Us

About

### FROM THE EXPERTS

Find Valuable Content At Reasonable Price

Explore Now





Login

OR

Sign Up

## (2) User Registration Page



### REGISTRATION

Name



Email Address

Password

Choose Avatar

Already Signed Up? [Login](#) here

## (3) Login Page



### Welcome to CourseBundler

Email Address

Password

Forget Password?

New User? [Sign Up](#) here

## (4) Profile Page



### PROFILE



[Change Photo](#)

**Name** Nitish Kumar

**Email** nitishgoswami9876@gmail.com

**CreatedAt** 2022-12-18

**Subscription** [Subscribe](#)

[Update Profile](#)

[Change Password](#)

## (5) Contact Page

### Contact Us

Name

Ritul Sharma

Email Address

ritul123@gmail.com

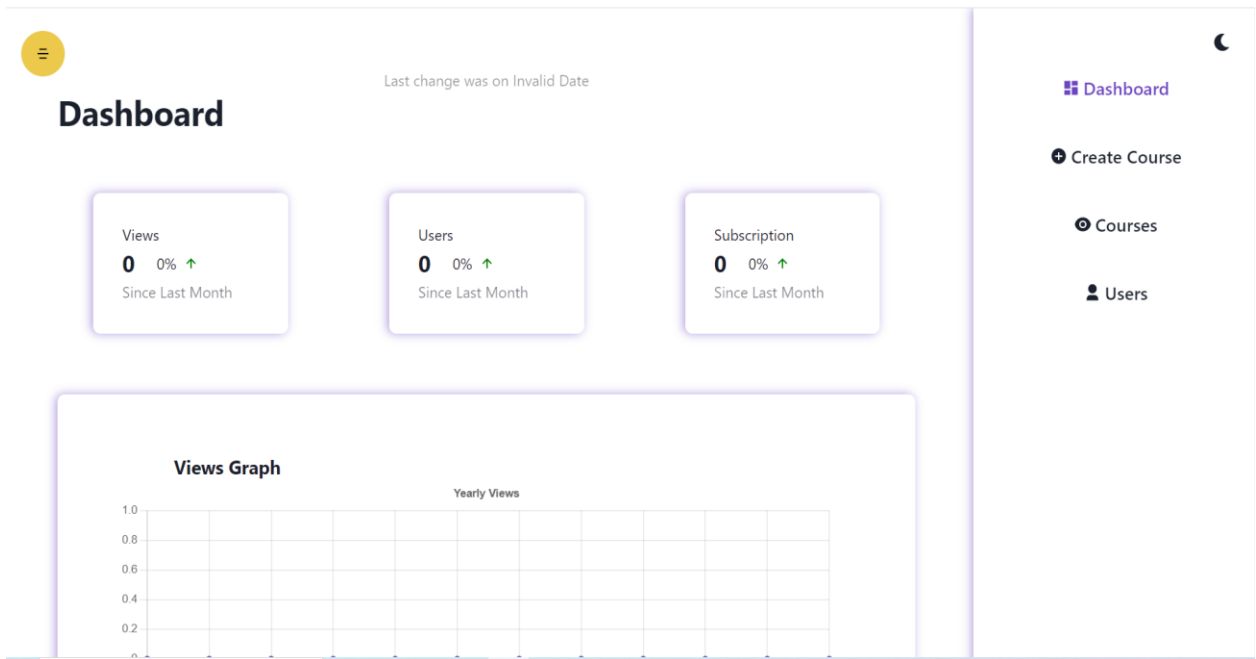
Message

Hi, Your course is awesome. I completed it but i have some question regarding certification. For how long it is valid

[Send Mail](#)

Request for a course? [Click](#) here

## (6)Admin Dashboard



## (7) Create Course

The "CREATE COURSE" form includes the following fields and buttons:

- Title
- Description
- Creator Name
- Category (dropdown menu)
- Choose File
- Create

⋮

ALL USERS

ID	NAME	EMAIL	ROLE	SUBSCRIPTION	ACTION
#639f1bcd18c1c19b65b908b	Nitish Kumar	nitishgoswami9876@gmail.com	admin	Not Active	<button>Change Role</button> <button></button>
#639f235cd717a41b9d370e	Roshan Kumar	roshan@gmail.com	user	Not Active	<button>Change Role</button> <button></button>
#639f2408096662450b6b8a5e	Ritul Sharma	ritul@gmail.com	user	Not Active	<button>Change Role</button> <button></button>
#63a138453e2243b9e4d8a4c0	Nikhil	nikhil@gmail.com	user	Not Active	<button>Change Role</button> <button></button>
#63a13a7badb01a1b30f8a8a8	Shobhit Goswami	shobhit@gmail.com	user	Not Active	<button>Change Role</button> <button></button>
#63a13d6be6ae7257ebd00838	Narender	narender@gmail.com	user	Not Active	<button>Change Role</button> <button></button>

All available users in the database

⚙️

Dashboard

Create Course

Courses

Users

## (8) Add Lecture

Web Development Bootcamp By Coding Ninja

X

### Web Development Bootcamp By Coding Ninja

#639f1f59d6f0e7a2ab7d970f

#### Lectures

##### #1 Introduction to Internet

Its your first intro to internet and you can clear all confusion about your internet

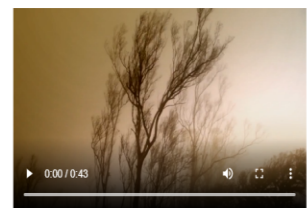


#### ADD LECTURE

Introduction to Internet

Its your first intro to internet and you can clear all confusio

Choose File



Upload

Close



ALL USERS

ID	NAME	EMAIL	ROLE	SUBSCRIPTION	ACTION
#639f1bcd18c1c19b65b908b	Nitish Kumar	nitishgoswami9876@gmail.com	admin	Not Active	<button>Change Role</button> <button></button>
#639f235cd717a41b99d370e	Roshan Kumar	roshan@gmail.com	user	Not Active	<button>Change Role</button> <button></button>
#639f240809662450b6b8a5e	Ritul Sharma	ritul@gmail.com	user	Not Active	<button>Change Role</button> <button></button>
#63a138453e2243b9e4d8a4c0	Nikhil	nikhil@gmail.com	user	Not Active	<button>Change Role</button> <button></button>
#63a13a7badb01a1b30f8a8a8	Shobhit Goswami	shobhit@gmail.com	user	Not Active	<button>Change Role</button> <button></button>
#63a13d6be6ae7257ebd00838	Narender	narender@gmail.com	user	Not Active	<button>Change Role</button> <button></button>

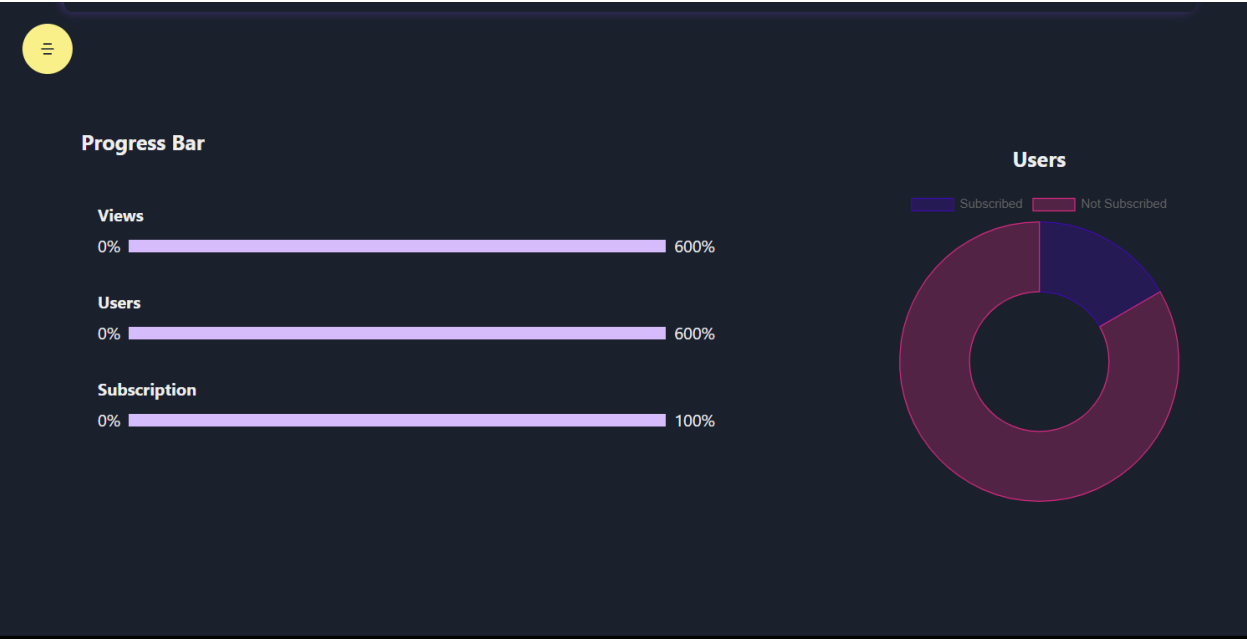
All available users in the database

Dashboard

Create Course

Courses

Users



## **Future scope**

Learn2Earn is major project and it require experstise in mentioned technology. But I devoted my major time towards learning tech-stack,

So due to limited time span there is lot of stuff to work on. Some of the major modules to work on are :

- Subscription Model
- UI/UX
- Automated reporting system.
- Course Recommendation.
- Integration of hiring platform.

## **References**

I learned all used technology through free resources available on internet. Mainly I followed official documentation for technology.

Other than this I used following resources in this project development Journey

- [Stackoverflow](#) (For doubt and error resolution)
- [Geeksforgeeks](#) (For understanding some concepts)
- [Pepcoding Web Dev. Course](#) (For learning MERN stack)