**FULL STACK DEVELOPMENT FOR**

**HIMALAYAN CHESS ACADEMY**

Tribhuvan University Institute of Science and Technology

Kathmandu, Nepal



An Internship Report Submission

In Partial Fulfillment of the Requirement for Degree of

Bachelor of Science in Computer Science and Information Technology

**Submitted To:**

National College of Computer Studies

Department of Computer Science and Information Technology

Paknajol, Kathmandu, Nepal

**Submitted By:**

Cyrus Maharjan (26312/077)

May 2025

**Under Supervision of**

Mr. Radha Krishna Gajurel

Lecturer

# SUPERVISOR’S RECOMMENDATION

I hereby recommend that this internship report has been prepared under my supervision by **Mr.** **Cyrus Maharjan (26312/077)** entitled "**ACADEMY MANAGEMENT SYSTEM**" in partial fulfilment of the requirements for the degree of Bachelor of Science in Computer Science and Information Technology to be processed for the evaluation.

……………………………….

**Signature of the Supervisor**

**Name: Mr. Radha Krishna Gajurel**

**Designation: Supervisor**

## ACKNOWLEDGEMENT

I would like to thank Himalayan Chess Pvt. Ltd. for giving me the opportunity to complete my internship and contribute to the development of a full-stack web application. I am especially grateful to my mentor, Mr. Dipesh Giri, and the entire team at the company for their guidance, support, and encouragement throughout the internship.

I also want to thank my college administration and teachers for their continued support. I am thankful to Tribhuvan University, Faculty of Science, for including the internship in the academic curriculum. This opportunity has allowed me to apply my knowledge in a real-world setting and gain valuable practical experience.

**Cyrus Maharjan**

**26312/077**

# ABSTRACT

This project documents an internship completed at Himalayan Chess Pvt. Ltd. where a full-stack web application was developed to digitize the academy’s administrative processes. The primary objective was to replace handwritten records related to student attendance, event management, and other administrative tasks with a user-friendly digital system. The application was built using Next.js, integrating both front-end and back-end components to create a fast, and efficient platform based on academy’s requirements. During the development, various challenges associated with transitioning from manual, paper-based processes to an electronic system were addressed through practical solutions. This report presents the overall development approach, key technologies employed such as cloud storage with Cloudinary and state management, and the positive impact of the digital transformation on operational efficiency. Additionally, the internship experience contributed significantly to professional growth in full-stack web development and modern application design.

**Keywords:** Next.js, Full-Stack Development, State Management, Digital Transformation, Cloudinary

## TABLE OF CONTENTS

[SUPERVISOR’S RECOMMENDATION ii](#_Toc198597456)

[ACKNOWLEDGEMENT iii](#_Toc198597457)

[ABSTRACT iv](#_Toc198597458)

[TABLE OF CONTENTS v](#_Toc198597459)

[LIST OF ABBREVIATIONS vi](#_Toc198597460)

[LIST OF TABLES vi](#_Toc198597461)

[CHAPTER 1 INTRODUCTION 1](#_Toc198597462)

[1.1 Introduction 1](#_Toc198597463)

[1.2 Problem Statement 2](#_Toc198597464)

[1.3 Objectives 2](#_Toc198597465)

[1.4 Scope and Limitation 3](#_Toc198597466)

[1.5 Report Organization 3](#_Toc198597467)

[CHAPTER 2 ORGANIZATION DETAILS AND LITERATURE REVIEW 5](#_Toc198597468)

[2.1 Introduction to Organization 5](#_Toc198597469)

[2.2 Organizational Hierarchy 6](#_Toc198597470)

[2.3 Working Domains of Organization 7](#_Toc198597471)

[2.4 Description of Intern Department/Unit 7](#_Toc198597472)

[2.5 Literature Review 9](#_Toc198597473)

[CHAPTER 3 INTERNSHIP ACTIVITIES 11](#_Toc198597474)

[3.1 Roles and Responsibilities 11](#_Toc198597475)

[3.2 Weekly log 11](#_Toc198597476)

[3.3 Description of the Project(s) Involved During Internship 14](#_Toc198597477)

[3.3.1 System Architecture and design 15](#_Toc198597478)

[3.3.2 Tools Used 16](#_Toc198597479)

[3.4 Activities Performed During Internship 16](#_Toc198597480)

[3.4.1 API Integration 16](#_Toc198597481)

[3.4.2 CRUD operation 17](#_Toc198597482)

[3.4.3 Cron Jobs Scheduler 19](#_Toc198597483)

[3.4.4 Send mail using nodemailer 20](#_Toc198597484)

[3.4.5 Middleware for server side role based authorization 21](#_Toc198597485)

[3.4.6 Upload file in cloudinary 21](#_Toc198597486)

[CHAPTER 4 CONCLUSION AND LEARNING OUTCOMES 23](#_Toc198597487)

[4.1 Conclusion 23](#_Toc198597488)

[4.2 Learning Outcomes 23](#_Toc198597489)

[REFERENCES 25](#_Toc198597490)

[APPENDIX](#_Toc198597491)

# LIST OF ABBREVIATIONS

**API** Application Programming Interface

**HTML** Hypertext Markup Language

**MVC** Model View Controller

**JSON** JavaScript Object Notation

**OTP** One-Time Password

**UTC** Coordinated Universal Time

**DB** Database

**HCA** Himalayan Chess Academy

**UI** User Interface

**UX** User Experience

# LIST OF TABLES

[Table 1: Overview of company profile 6](#_Toc198597428)

[Table 2 :Overview of Internship. 8](#_Toc198597429)

# CHAPTER 1 INTRODUCTION

## 1.1 Introduction

As the need for better management of academy operations continued to grow, a full-stack web application was developed for Himalayan Chess Pvt. Ltd. to improve the efficiency of administrative tasks. The project’s primary goal was to replace the outdated manual, paper-based systems with a modern digital solution, providing a streamlined platform for tracking student attendance, organizing events, and managing day-to-day academy activities.

The application serves as a centralized system where users can easily manage student records, monitor attendance, and schedule events. Key features include automated attendance tracking, activity record scheduling, and real-time notifications to keep users updated. By moving these tasks online, the system reduces the manual effort involved in administration and allows staff to focus more on other important areas of the academy.

Additionally, the platform includes role-based access control, ensuring that only authorized users can access specific features or data. This helps maintain privacy and security while allowing different users to interact with the system according to their role. The use of Next.js for development helped ensure the platform is scalable, responsive, and performs efficiently across devices, providing a seamless user experience.

Security and data protection were major priorities during development. Features like secure authentication, OTP, and periodic backups were implemented to safeguard sensitive information. This web application not only replaced the academy's traditional handwritten systems but also aimed to enhance overall productivity, improve communication, and reduce administrative errors, ultimately contributing to a more organized and efficient environment at Himalayan Chess Pvt. Ltd.

## 1.2 Problem Statement

Efficient management of administrative tasks can be challenging for organizations, especially when paper-based systems are used. In traditional methods, attendance is recorded manually, events are organized through handwritten notes, and important information is stored on paper. These outdated practices often result in confusion, errors, and inefficiencies. Quick access to and analysis of data become difficult, leading to wasted time and lower productivity.

To address these issues, a web application was developed to reduce the dependency on paper-based systems and to make it easier to access, add, store, and analyze data. Through this digital platform, administrative tasks—such as tracking student attendance and scheduling events—can be managed in a single system. The use of **Next.js** allowed a responsive and user-friendly interface to be built, offering a smooth experience across various devices.

While developing the application, several challenges were encountered, including secure data storage, role-based access for different users, and the need to keep the system simple and easy to use. Solutions to these challenges are explored in this project, with the aim of creating a platform that simplifies data management and enhances overall efficiency at the academy.

## 1.3 Objectives

The key objectives of this internship are:

* To develop a full-stack web application for Himalayan Chess Pvt. Ltd. to efficiently manage student records, attendance, activity records, events, etc.
* To implement role-based access control with three user roles: Superadmin, Admin, and Trainer, ensuring each role has appropriate permissions and access to features based on their responsibilities.
* To integrate real-time notifications for updates on activity records, leave requests, and other important academy activities to keep users informed and engaged.

## 1.4 Scope and Limitation

**Scopes:**

* Develop a web application for Himalayan Chess Academy to manage student records, attendance, and events in an easy way.
* Provide full access to CRUD (Create, Read, Update, Delete) operations for the Superadmin, allowing them to manage all data and users.
* Offer limited features for Admin and Trainer roles, enabling them to manage student attendance and events according to their role.
* Ensure the system is secure and simple to use, with each user role having access to the appropriate information.

**Limitations:**

* The application might be difficult for non-technical users to fully navigate without some training or guidance.
* Some features might require users to understand basic concepts of using a web-based system, which could be challenging for those unfamiliar with technology.
* Due to the non-technical nature of some users, there may be a learning curve, and additional support may be needed to ensure smooth adoption of the system.

## 1.5 Report Organization

* **Chapter 1: Introduction**  
  Provides an overview of the internship project, including the problem statement, objectives, scope, limitations, and structure of the report.
* **Chapter 2: Organization Details and Literature Review**  
  Describes the organization where the internship was completed. It includes the organizational structure, working domains, the department/unit where the internship was carried out, and a review of related studies.
* **Chapter 3: Internship Activities**  
  Details the intern's roles and responsibilities, weekly log of tasks, description of the projects involved, and the technical activities performed during the internship.
* **Chapter 4: Conclusion and Learning Outcomes**

Summarizes the overall experience of the internship and highlights the key skills and knowledge gained throughout the period.

# CHAPTER 2 ORGANIZATION DETAILS AND LITERATURE REVIEW

## Introduction to Organization

Himalayan Chess Pvt. Ltd. established in 2023, is an institution in Nepal dedicated to promoting and teaching chess. The academy was created to make chess accessible to everyone, from beginners to advanced players. With a welcoming and supportive environment, it aims to help students improve their chess skills while having fun. Also, with the guidance of experienced instructors and the leadership of its CEO, Monalisha Khamboo, the academy creates a friendly place where students can grow their skills.

The academy offers various courses designed for all skill levels, from those just starting out to experienced players. With experienced instructors, students receive the guidance and support they need to grow in their chess journey. The academy focuses on developing strategic thinking, problem-solving skills, and a love for the game.

Himalayan Chess Academy also fosters a strong sense of community by organizing events and activities that bring students together to learn, play, and share their experiences. This creates a supportive environment where students can build friendships and stay motivated.

Overall, Himalayan Chess Academy is committed to providing a fun and engaging way to learn chess, while encouraging growth, learning, and teamwork. It’s the perfect place for anyone who wants to explore chess, no matter their experience level.



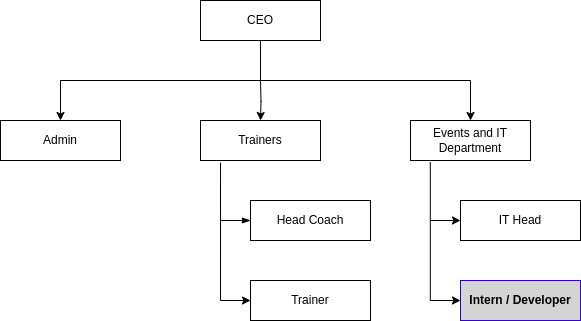
**Figure** **2.1:** Organization Logo

Table : Overview of company profile

|  |  |
| --- | --- |
| Organization Name | Himalayan Chess Private Limited |
| Address | Thapathali, Kathmandu, Nepal |
| Contact | +977 9849264344 |
| Year of establishment | 2023 |
| Email | [hello@himalyanchess.com](mailto:hello@himalyanchess.com) |
| Website | [https://himalayanchess.com](https://himalayanchess.com/) |

## Organizational Hierarchy

The organizational structure of Himalayan Chess Academy has been designed to ensure efficient operation and coordination among various departments. The structure promotes effective communication and teamwork across all levels.



**Figure 2.2** Organization Hierarchy Chart

## Working Domains of Organization

Himalayan Chess Academy focuses on a variety of services to enhance the learning experience for its students and improve its overall operations. The main working domains of the academy includes:

* **Chess Lessons to Students:** The academy provides structured chess training to students of all levels, helping them develop their skills, strategic thinking, and problem-solving abilities through personalized lessons.
* **Event Management**: The academy organizes chess tournaments, competitions, and special events, creating opportunities for students to challenge themselves, engage with others, and showcase their skills in a competitive environment.
* **Graphic Design for social media**: The academy utilizes graphic design to create engaging posts for its social media platforms. These designs help promote upcoming events, share important updates, and enhance the academy's online presence.
* **Software Development for Academy System**: The academy develops and maintains its internal systems, including a student management system, event tracking, and activity management. Software development ensures smooth operations and better management of academy activities.

## Description of Intern Department/Unit

The Software Development Unit at Himalayan Chess Academy is an important part of the organization that focuses on building and managing all the digital systems used by the academy. This includes creating software for managing student information, keeping records of their activities, attendance records, payment records, and other internal systems that help the academy work more efficiently. The software development unit handles both the frontend and the backend part. The goal of the department is to make the daily operations of the academy smoother, more organized, and user-friendly through well-designed software solutions.

As an intern in this department, tasks were carried out under the supervision of the IT Head, who also acted as a mentor. There was a strong focus on learning and growing technical skills as well as non-technical skills. The work environment was friendly and supportive, with team members always willing to help and provide feedback. This made the department a great place to develop both technical knowledge and practical experience in real-world software development.

Table :Overview of Internship.

|  |  |
| --- | --- |
| **Start Date** | 14 February, 2025 |
| **Total Duration** | 3 Months |
| **Working Days** | 5 days a week |
| **Intern Position** | Full Stack Developer |
| **Mentor** | Dipesh Giri |
| **Office Hour** | 10:00 AM to 5:00 PM |

## 2.5 Literature Review

This section reviews the existing literature and technologies relevant to the development of a web-based academy management system, focusing on modern web frameworks, database management, authentication methods, and automation techniques.

**Web Frameworks: Next.js**

Next.js is a popular React-based framework designed to build scalable web applications with support for both client-side and server-side rendering. It simplifies routing, API development, and enables serverless functions, which makes it well-suited for full-stack applications. According to Vercel, Next.js improves performance and SEO by rendering pages on the server, while also providing flexibility with static site generation and incremental static regeneration (Next.js Documentation, 2023).

**Database Management: MongoDB**

MongoDB is a NoSQL database that stores data in flexible, JSON-like documents. This design allows for easy changes to the data structure, making it ideal for applications that need to adapt quickly, such as academy management systems. MongoDB supports powerful querying, indexing, and can scale horizontally to handle large amounts of data efficiently (Ozer, 2021).

**State Management: Redux**

Redux is a tool used with React to manage the global state of an application in a predictable way. It helps keep the app’s data organized and makes it easier to find and fix issues. Redux improves how data flows between parts of the app, which is important for apps that update in real time or have many users (Abramov, 2015).

**Authentication and Authorization: NextAuth.js**

NextAuth.js is an open-source authentication tool made for Next.js applications. It provides secure user login and role-based access control, helping developers manage user sessions easily. Using secure authentication like this is important to protect sensitive academy data and make sure only authorized users can access certain features.

**Automation: Vercel Scheduler and Cron Jobs**

Scheduling automated tasks like sending notifications or performing maintenance can be done using cron jobs. Vercel Scheduler is a serverless cron job service that works well with Next.js apps, letting developers run functions at set times . Automating tasks such as birthday notifications helps keep users engaged and lowers manual work.

**Email Notifications: Nodemailer**

Nodemailer is a module for Node.js that makes sending emails easy and reliable ( Nodemailer Documentation., 2023). It supports different ways to send emails, like SMTP, and can be integrated into apps to send notifications to trainers or students in an academy system.

**School and Academy Management Systems**

Using digital systems in education streamlines administrative tasks, reduces errors, and improves efficiency. Web-based management systems typically include features like managing student records, tracking attendance, scheduling events, and handling fees. These systems also enhance communication between staff and students, fostering a better learning environment (OECD, 2023).

# CHAPTER 3 INTERNSHIP ACTIVITIES

## Roles and Responsibilities

The role of Full-Stack Developer was assigned during the internship at Himalayan Chess Academy. The main responsibility was to design and develop a fully functional web application aimed at digitizing the academy’s administrative processes.

My responsibilities included:

* UI/UX design principles were researched, and a custom interface was created based on the academy’s requirements.
* Secure authentication and role-based authorization were implemented using NextAuth.js.
* API routes were developed to handle backend operations, such as managing academy records.
* A backend database was integrated to store and manage application data efficiently.
* A cron job was configured using Vercel Scheduler to automatically trigger an API for sending birthday notifications each week.
* Various filtering options were implemented to allow users to easily search and filter records based on different parameters.

## 3.2 Weekly log

My intern duration was 3 months starting from February 14, 2025, to May 14, 2025. During this time, various tasks were performed, as detailed in the weekly log provided below.

|  |  |
| --- | --- |
| **Week** | **Areas** **of** **Activities** |
| 1 | * Gathered detailed requirements related to attendance tracking student data management. * Designed simple schemas with primary required fields. * Implemented login and logout operation for the system. * Made simple prototype design. |
| 2 | * CRUD Operation from users, batches, and projects. * Refreshed my understanding of NextAuth.js concepts for implementing secure authentication and authorization. * Implemented server-side role-based authorization using next-auth and next.js middleware. |
| 3 | * CRUD for students, leave request. * Sidebar design using framer motion. * Designed activity record model with required fields. * Started assign class to trainer module. |
| 4 | * Designed activity record model with required fields. * Started assign class to trainer module. * Started setting up Redux for state management, including installing the necessary packages and creating thunks and reducers for handling asynchronous actions. |
| 5 | * Learned about date time conversion from UTC to our local time zone using dayjs library. * Created a calendar to assign the class for future days as well. * Linked activity records to trainers. And update assigned classes. |
| 6 | * Performed final testing and debugging * Documented project flow and key learnings from the development process. |
| 7 | * CRUD for courses with JSON file upload feature. * File upload for trainers CV, leave support file, etc using Cloudinary. * Created loading component to show in initial page load. |
| 8 | * Send email to trainers, admin and superadmins about the assigned class with details. * Converted most of the modal designs to their separate page. * Export the filtered activity records to excel sheet. * Learned about vercel scheduler (Cron Jobs) for automating tasks like weekly birthday notifications. |
| 9 | * Added sub menus in view students, users and batches component for easy access. * Formatted email HTML contents to have all the required details in proper format. * Added profile images to students with edit feature. * Created CRUD operations for study materials. |
| 10 | * Created trainer side class activity record update module. * Created Trainer side leave request and send email to admin and superadmin. * Designed new attendance approach with new schema design. * Used redux to properly fetch the selected classes students and other records. |
| 11 | * Included a branch feature of the academy. * Granted only branch access to branch admin and full privileges to global admin and superadmin. * CRUD for test histories of students and linked to respective view student module. |
| 12 | * Designed Payment record schema. * Started payment module for tracking incoming and outgoing payments for academy. * Added payment files and linked to payment using Cloudinary. |
| 13 | * Designed different tournaments record schema. * CRUD for lichess tournaments and other tournaments. * CRUD for HCA Circuit tournaments and their corresponding series. * Linked tournaments to respective view student page. |

## 3.3 Description of the Project(s) Involved During Internship

The academy management system is designed to manage all administrative workflows through a simple digital platform. All handwritten and manual tasks are reduced through its implementation. Both frontend and backend integrations required for the software have been included in the project.

The tasks are mentioned below:

* CRUD operations were implemented for all records to enable easy creation, reading, updating, and deletion of data.
* Database schemas were designed for all models, and custom APIs were created to handle various backend operations.
* Filtering options were provided through multiple parameters to allow efficient search and navigation of records.
* Class assignments were managed for trainers, and email notifications were sent using Nodemailer to inform them of their schedules.
* Attendance records were maintained for all users of the academy, including students and trainers.
* A leave request system was developed where users could apply for leave, and approval status was managed by the admin.
* Payment records were maintained, including both incoming fees from students and outgoing expenses of the academy.
* A tournament management module was added, where tournament records such as participants, match details, and results were stored and managed.
* Birthday notifications were automated and sent weekly using a cron job configured through Vercel Scheduler.
* Role-based access control was enforced to ensure users accessed only the features relevant to their roles.

### 3.3.1 System Architecture and design

The system follows a basic Model-View-Controller (MVC) architecture to keep the code organized and modular. The Model handles the database schemas and defines how data like students, attendance, and events are stored and managed. The View is built using React and Tailwind CSS, responsible for displaying the user interface in a clean and responsive way. The Controller is made up of API routes in Next.js, which handle user actions, communicate with the database, and return data to the frontend. Using this approach made it easier to manage different parts of the application and improved code readability and scalability.

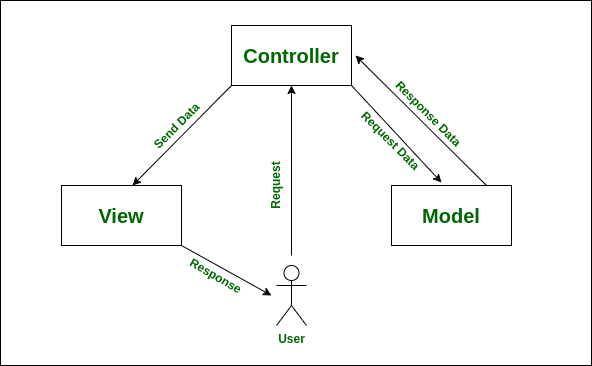


Fig 1: MVC System Architecture

### 3.3.2 Tools Used

In the implementation of the system, a variety of tools and technologies were used to facilitate the development process and ensure the successful creation of the system. The following tools were utilized:

**Front end tools**

* **React.js** to build dynamic and component-based user interfaces.
* **Tailwind CSS** for fast and responsive styling using utility-first classes.
* **Redux** for managing global state across the application, especially for handling user and attendance data.
* **Material UI** to provide pre-built, accessible components for a consistent and modern design.

**Backend tools**

* **Node.js** as the runtime environment to build the server-side logic of the application.
* **Express.js** as the web framework to handle routing and API requests efficiently.
* **MongoDB** as the NoSQL database to store and manage data for students, attendance, and events.

**Version control tools**

* **Git** for version control to track changes and collaborate effectively.
* **GitHub** for storing code in repository and managing project collaboration.

**Development tools**

* **Vercel** for seamless deployment of the application.

## 3.4 Activities Performed During Internship

### 3.4.1 API Integration

APIs were integrated to connect the frontend and backend of the application, enabling dynamic data retrieval and updates. Next.js API routes were used to handle requests such as fetching student information and submitting attendance records. These routes were configured to communicate with the MongoDB database for storing and retrieving data, ensuring that user interactions on the frontend were accurately reflected in real-time on the server. To manage application state efficiently, Redux was utilized to fetch data from the APIs and store it in the global state. This allowed data to be accessed across various components in the frontend, ensuring a smooth and consistent user experience.

export const fetchAllBatches = createAsyncThunk(

"batches/getAllBatches",

async (\_, { rejectWithValue }) => {

try {

const response = await fetch("/api/batches/getAllBatches");

const resData = await response.json();

return resData.allBatches;

} catch (error: any) {

return rejectWithValue(error.message);

}

}

); /// extra reducer

.addCase(fetchAllBatches.pending, (state) => {

state.allBatchesLoading = true;

})

.addCase(fetchAllBatches.fulfilled, (state, action) => {

state.allBatches = action.payload?.sort(

(a: any, b: any) =>

new Date(b.createdAt).getTime() - new Date(a.createdAt).getTime()

); })

### 3.4.2 CRUD operation

Create operation is implemented when a new record is added. It involves sending a POST request to the backend API to store the new data in the MongoDB database. After the data is successfully stored, the frontend is updated by dispatching the action to Redux to reflect the new record.

export async function POST(request: NextRequest) {

try {

const timeZone = "Asia/Kathmandu";

await dbconnect();

const reqBody = await request.json();

const newAssignClass = new ActivityRecord({

...reqBody,

nepaliDate: passedNepaliDate.format(),

utcDate: convertedUtcDate,

weekNumber,

weekStartDate,

weekEndDate,

});

const savedNewAssignClass = await newAssignClass.save();

if (savedNewAssignClass) {

await sendAssignClassMail({

subject: "Class assignment to trainer",

assignedClass: reqBody,

});

return NextResponse.json({

statusCode: 200,

msg: "Class Assigned",

savedNewAssignClass,

});

} }}

Read operation retrieves data from the database. A GET or POST request is sent to the backend API, and the data is then stored in the Redux state to be used by the frontend.

export async function GET(request: NextRequest) {

try {

await dbconnect();

const allBatches = await Batch.find({});

if (allBatches) {

return NextResponse.json({

msg: "All batches found",

statusCode: 200,

allBatches,

});

}

}

}

Update operation is used to modify existing records. A PUT or PATCH request is sent to the backend API with the updated data, and the Redux store is updated to reflect the changes in the frontend. I used POST method for update as well.

const updatedBatch = await Batch.findOneAndUpdate(  
 { \_id: reqBody.\_id },  
 {  
 ...reqBody,  
 batchStartDate: utcBatchStartDate,  
 batchEndDate: utcBatchEndDate,  
 }  
);  
if (updatedBatch) {  
 return NextResponse.json({  
 msg: "Batch updated",  
 statusCode: 200,  
 updatedBatch,  
 });  
}

### 3.4.3 Cron Jobs Scheduler

Created an api as “checkUsersBirthdays” to get all the birthdays of users in this week and send email to admin and superadmin.

{ "crons": [ { "path": "/api/cron/checkUsersBirthdays", "schedule": "15 0 \* \* 0" } ] }

And for checkUsersBirthdays api,

const birthdayPeople = allPeople.filter((person) => {

if (!person.dob) return false;

const dob = dayjs(person.dob).tz("Asia/Kathmandu");

const dobThisYear = dob.year(today.year());

return (

dobThisYear.isSameOrAfter(startOfWeek, "day") &&

dobThisYear.isSameOrBefore(endOfWeek, "day")

);

});

await sendBirthdayMail({

subject: "🎉 Users and Students with Birthdays This Week!",

birthdayPeople,

weekRange: `${startOfWeek.format("MMMM D")} - ${endOfWeek.format(

"MMMM D"

)}`, });

return Response.json({

msg: `Sent birthday email for ${birthdayPeople.length} person(s).`,

statusCode: 200,

});

### 3.4.4 Send mail using nodemailer

export async function sendOtpMail({ otp, email, subject }: any) {

try {

const transporter = nodemailer.createTransport({

service: "gmail",

auth: {

user: process.env.GMAIL\_EMAIL\_ADDRESS,

pass: process.env.GMAIL\_APP\_PASSWORD,

},

});

const options = {

from: process.env.GMAIL\_EMAIL\_ADDRESS, // sender address

to: email,

subject,

html: getOTPEmailContent(otp),

};

const info = await transporter.sendMail(options);

console.log("OTP email sent successfully");

return info;

} catch (error) {

console.log("Error sending otp email", error);

}}

### 3.4.5 Middleware for server side role based authorization

export async function middleware(request: NextRequest) {

const path = request.nextUrl.pathname;

const session = await getCurrentUser();

if (authRoutes.includes(path)) {

if (session?.user?.role) {

const role = session.user.role.toLowerCase();

return NextResponse.redirect(new URL(`/${role}/dashboard`, request.url));

}

return NextResponse.next();

}

}

### 3.4.6 Upload file in cloudinary

export async function POST(request: NextRequest) {

try {

const formData = await request.formData();

const file = formData.get("file") as File | null;

const fileType = file?.type.startsWith("image/") ? "image" : "raw";

const folderName = formData.get("folderName") as string;

let cloudinaryFileType = formData.get("cloudinaryFileType") as string;

const selectedConfig =

cloudinaryConfigs[cloudinaryFileType as keyof typeof cloudinaryConfigs];

// can have these values

// ["profileImage","studyMaterials","otherFiles","paymentFiles"]

if (!cloudinaryFileType) {

cloudinaryFileType = "otherFiles";

}

cloudinary.config(selectedConfig);

const bytes = await file.arrayBuffer();

const buffer = Buffer.from(bytes);

const res = await new Promise<cloudinaryResut>((res, rej) => {

const uploadStream = cloudinary.uploader.upload\_stream(

{

resource\_type: fileType, // This ensures PDF files are handled correctly

folder: folderName,

},

(error, result) => {

if (error) rej(error);

else res(result as cloudinaryResut);

}

);

uploadStream.end(buffer);

});

console.log("file upload response ", res);

return NextResponse.json({

msg: "File uploaded",

res,

statusCode: 200,

});

}

# CHAPTER 4 CONCLUSION AND LEARNING OUTCOMES

## 4.1 Conclusion

During the internship at Himalayan Chess Academy, valuable hands-on experience was gained in developing a full-stack web application that aimed to simplify and digitize the academy’s daily operations. The work involved both frontend and backend development, where various important features were created, such as user login and authentication, managing student records, keeping attendance, scheduling events, and automating tasks like sending birthday notifications. This helped in understanding how different parts of a web application work together to create a smooth and user-friendly system.

This internship not only helped in improving technical skills but also played an important role in building soft skills like communication, teamwork, and time management. Talking to non-technical users and explaining technical features in a simple way made communication clearer. Working in a real environment with real users also gave a better understanding of how software is developed, tested, and delivered in professional settings. Overall, the internship was a great learning experience that provided a strong foundation for future projects and real-world software development.

## 4.2 Learning Outcomes

During the internship period, several technical and non-technical skills were learned and developed:

* A clear understanding of full-stack web development was gained, including how frontend and backend work together in a complete application.
* Practical experience was gained in using Next.js for building frontend and backend routes, MongoDB for managing data, and Redux for handling global application state.
* Skills in creating and using APIs were developed to handle actions like submitting attendance, managing students, assigning classes, and sending notifications.
* Experience was gained in building a user-friendly interface by applying UI/UX design principles that made the system easy to use for the academy staff.
* Real-time communication with mentors and users improved the ability to collect feedback and update the project to better meet user needs.
* Authentication and role-based authorization were implemented using NextAuth.js, which added security and control to the system.
* Weekly automated tasks, such as sending birthday emails, were handled using Vercel Scheduler, and emails were sent using Nodemailer, helping to understand automation tools.
* Knowledge of how to manage a software project, from planning to delivery, was improved through active involvement in all stages of development.
* Problem-solving and debugging skills were sharpened by fixing errors, testing the system, and making sure everything worked smoothly.
* Communication and collaboration were improved by working closely with the academy staff, understanding their needs, and explaining technical features in simple terms.

This internship experience helped build a strong foundation not just in coding, but also in teamwork, communication, and real-world software development. It gave the confidence to take on more complex projects in the future.

# REFERENCES

Nodemailer Documentation. (2023). Retrieved from https://nodemailer.com/about/

Abramov, D. &. (2015). Redux. Retrieved from https://redux.js.org/

Next.js Documentation. (2023). Retrieved from https://nextjs.org/docs

OECD. (2023). Education and student information systems. Retrieved from https://www.oecd.org/en/publications/oecd-digital-education-outlook-2023\_c74f03de-en/full-report/education-and-student-information-systems\_ef9f7b25.html

Ozer, C. B. (2021). Introduction to MongoDB. Retrieved from https://medium.com/codex/introduction-to-mongodb-16098b30d32b

# APPENDIX

