ABSTRACT

Task management systems are pivotal in orchestrating and prioritizing tasks within projects or organizations, enhancing productivity and organization. The development of a comprehensive task management system within the MERN (MongoDB, Express.js, React.js, Node.js) stack entails integrating various components to facilitate efficient task creation, assignment, tracking, and completion.

The task management system encompasses key features such as user authentication, task creation, assignment, tracking, notifications, search and filtering, and reporting and analytics. Users can securely authenticate and access the system, create tasks with detailed information, assign tasks to specific users or teams, track task progress, receive notifications about assignments and updates, and utilize search and filtering functionalities to quickly find relevant tasks. Furthermore, the system offers reporting and analytics features for analyzing task performance and optimizing workflows.

In this project, the architecture of the system involves leveraging React.js for the frontend, Node.js with Express.js for the backend, and MongoDB for the database. React.js offers a dynamic and responsive user interface, while Node.js with Express.js establishes a RESTful API to handle client requests and interact with the database. MongoDB, known for its flexibility and scalability, serves as the database for storing task-related data, ensuring seamless data flow through HTTP requests between the frontend and backend.The development approach involves adopting Agile methodologies for iterative development, utilizing version control systems for collaborative development and code management, implementing automated testing for reliability, and following best practices for documentation, code reviews, and deployment processes.

CHAPTER 1

INTRODUCTION

Purpose:

1. Organization: Task management helps individuals or teams to organize their work efficiently. By breaking down larger projects into smaller tasks, it becomes easier to manage and prioritize them effectively.

2. Prioritization: It enables individuals or teams to prioritize tasks based on deadlines, importance, or dependencies. This ensures that the most critical tasks are addressed first, reducing the risk of missing deadlines or neglecting important work.

3. Time Management: Task management assists in allocating time effectively. By setting deadlines and estimating the time required for each task, individuals can better manage their schedules and ensure that tasks are completed on time.

4. Progress Tracking: It allows individuals or teams to track the progress of tasks and projects. This helps in identifying bottlenecks, adjusting timelines, and ensuring that projects stay on track towards completion.

5. Collaboration: Task management tools often facilitate collaboration among team members by allowing them to assign tasks, share updates, and communicate effectively. This promotes teamwork and ensures that everyone is on the same page regarding project goals and progress.

6. Resource Allocation: Task management helps in allocating resources such as manpower, budget, and equipment efficiently. By having a clear overview of all tasks and their requirements, organizations can optimize resource utilization and avoid overloading individuals or departments.

7. Decision Making: Effective task management provides valuable

1.1 PROJECT INTRODUCTION

1. Project Owner/Manager: Identify the individual or team responsible for overseeing the project. This person serves as the main point of contact and is accountable for its successful execution.
2. Project Timeline: Outline the project's timeline, including start and end dates, as well as any important milestones or deadlines. This helps team members understand the project's timeline and allocate resources accordingly.
3. Project Goals and Objectives: Clearly define the goals and objectives that the project aims to achieve. This provides clarity on what success looks like and helps keep the project focused on delivering value to stakeholders.
4. Scope and Deliverables: Describe the scope of the project and list the deliverables or outcomes that are expected upon completion. This ensures that everyone involved understands what needs to be accomplished and helps manage expectations.
5. Key Tasks and Activities: Highlight the key tasks and activities that need to be completed to achieve the project's objectives. This may include milestones, dependencies, and critical path activities.
6. Resource Allocation: Specify the resources required for the project, including human resources, budget, and materials. This helps ensure that adequate resources are allocated to support project activities.
7. Risk Assessment: Identify potential risks and uncertainties that may impact the project's success, along with mitigation strategies to address them. This helps proactively manage risks and minimize their impact on the project.
8. Communication Plan: Outline the communication plan for the project, including channels, frequency, and stakeholders involved. Effective communication is critical for keeping team members informed and aligned throughout the project lifecycle.

CHAPTER 2

WORKING ENVIRONMENT

The system is tailored to operate within a dynamic workplace environment, providing seamless task management capabilities to enhance productivity and collaboration among teams. Key features are integrated to optimize usability and accessibility for a diverse user base.

2.1 HARDWARE REQUIREMENT

CPU-intensiveness	Average intensity
Processor	12th Gen Intel(R) Core(TM) i5-1240P 1.70 GHz
RAM Size	16 GB

Table 2.1

2.2 SOFTWARE REQUIREMENT

Operating system	Windows 11
Frontend	HTML,CSS,JAVASCRIPT,REACTJS
Software Using	VS CODE,MONGODB COMPASS, POSTMAN
Backend	NODEJS,EXPRESSJS
Database	MONGODB
Documentation	MS - Office

Table 2.2

2.3 SYSTEM SOFTWARE

5.2.1 MERN(MongoDB,ExpressJS,ReactJS,NodeJS):

- MERN is an acronym that stands for MongoDB, Express.js, React, and Node.js. It represents a full-stack JavaScript framework used for building dynamic web applications. Each component in the MERN stack plays a specific role.
- MERN stack is a collection of technologies that enables faster application development. It is used by developers worldwide. The main purpose of using MERN stack is to develop apps using Javascriptonly. This is because the four technologies that make up the technology stack are all JS-based. Thus, if one knows JavaScript (and JSON), the backend, frontend, and database can be operated easily.

5.2.2 MERN STACK COMPONENTS:

There are four components of the MERN stack. Let's discuss each of them one by one.

- The first component is MongoDB, which is a NoSQL database management system.
- The second MERN stack component is ExpressJS. It is a backend web application framework for NodeJS.
- The third component is ReactJS, a JavaScript library for developing UIs based on UI components.
- The final component of the MERN stack is NodeJS. It is a JS runtime environment, i.e., it enables running JavaScript code outside the browser.

- **MongoDB:**

MongoDB is an open-source database that uses a document-oriented data model and a non-structured query language. It is one of the most powerful NoSQL systems and databases around, today. MongoDB Atlas is a cloud database solution for contemporary applications that is available globally. This best-in-class automation and established practices offer to deploy fully managed MongoDB across AWS, Google Cloud, and Azure.

It also ensures availability, scalability, and compliance with the most stringent data

security and privacy requirements. MongoDB Cloud is a unified data platform that includes a global cloud database, search, data lake, mobile, and application services.The basic unit of data in this database consists of a set of key-value pairs. It allows documents to have different fields and structures. This database uses a document storage format called BSON which is a binary style of JSON documents.

- **Features in MongoDB**

There are many features in MongoDB, some of which are discussed below

- Full cloud-based developer data platform
- Flexible document schemas
- Widely supported and code-native data access
- Change-friendly design
- Powerful querying and analytics
- Easy horizontal scale-out with sharding
- Simple installation
- Cost-effective
- Full technical support and documentation

- **ExpressJS:**

ExpressJS functions as a vital bridge connecting the front-end and back-end elements of a web application. Acting as an intermediary, it facilitates seamless real-time interaction between users and the application by efficiently processing and transmitting data. This framework offers a diverse range of beneficial features, such as middleware, routing, and templating, which greatly streamline the process of developing and maintaining web applications.

Technically, ExpressJS is an open-source web application framework, freely available, and built upon the Node.js platform. Its primary objective is to streamline the creation and management of server-side applications. With ExpressJS, you can effortlessly construct dynamic and interactive websites, establish RESTful APIs, and efficiently handle HTTP requests and responses.

ExpressJS caters to the needs of both novice and seasoned web developers. Its straightforward yet robust architecture makes it user-friendly and easily graspable, while its extensive feature set renders it suitable for developing intricate web applications.

- **Features in ExpressJS**

There are many features in ExpressJS, some of which are discussed below

- Routing Capabilities.
- Express Middleware functionality.
- Templating Engine Support.
- MVC Architecture Support.
- Easy and Flexible API Development.
- Error Handling Abilities.
- JSON Web Token (JWT) Integration.
- Encourage Scalability and Performance.

- **ReactJS**

React.js, more commonly known as React, is a free, open-source JavaScript library. It works best to build user interfaces by combining sections of code (components) into full websites. Originally built by Facebook, Meta and the open-source community now maintain it. One of the good things about React is that you can use it as much or as little as you want! For example, you can build your entire site in React or just use one single React component on one page.

React.js is built using JSX – A combination of JavaScript and XML. Elements are created using JSX, then use JavaScript to render them on your site. While React has a steep learning curve for a junior developer, it's quickly shaping into one of the most popular and in-demand JavaScript libraries.

React is considered a JavaScript library rather than a framework, whereas the other options we'll consider today are considered frameworks. It helps to think of a library as a tool that developers could use in any project and a framework as a whole design.

- **Features in ReactJS**

  There are many features in ReactJS, some of which are discussed below

  - Declarative: Simplifies UI development by describing how the UI should look at any given time.
  - Component-Based Architecture: Encourages the creation of reusable, self-contained UI components.
  - Virtual DOM: Improves performance by minimizing DOM manipulation through a lightweight virtual representation of the DOM.
  - JSX: Allows writing HTML-like syntax within JavaScript, enhancing code readability and maintainability.
  - One-Way Data Binding: Ensures predictable data flow by passing data down from parent to child components.
  - Lifecycle Methods: Offers lifecycle hooks for managing component initialization, updates, and unmounting.
  - Unidirectional Data Flow: Facilitates easier debugging and understanding of data flow in complex applications.
  - Server-Side Rendering (SSR): Supports rendering React components on the server, improving initial page load times and SEO.
  - Community and Ecosystem: Benefits from a vast community and ecosystem with libraries, tools, and resources for various needs.
  - Accessibility (A11y) Support: Provides built-in features and guidelines to ensure React applications are accessible to users with disabilities.

- **NodeJS**

Node.js is an open-source and cross-platform JavaScript runtime environment. Node.js runs the V8 JavaScript engine, the core of Google Chrome, outside of the browser. This allows Node.js to be very performant.

A Node.js app runs in a single process, without creating a new thread for every request. Node.js provides a set of asynchronous I/O primitives in its standard library that prevent JavaScript code from blocking and generally, libraries in Node.js are written using non-blocking paradigms, making blocking behavior the exception rather than the norm.

When Node.js performs an I/O operation, like reading from the network, accessing a database or the filesystem, instead of blocking the thread and wasting CPU cycles waiting, Node.js will resume the operations when the response comes back.This allows Node.js to handle thousands of concurrent connections with a single server without introducing the burden of managing thread concurrency, which could be a significant source of bugs.

Node.js has a unique advantage because millions of frontend developers that write JavaScript for the browser are now able to write the server-side code in addition to the client-side code without the need to learn a completely different language.

- **Features in NodeJS**

  There are many features in NodeJS, some of which are discussed below

  - Single Threaded Language
  - Asynchronous.
  - Event Driven.
  - Open Source.
  - High Performance.
  - Highly Scalable.
  - Node Package Manager(NPM)
  - No Buffering.
  - High Scalability

- Seamless JSON support

- **Visual Studio Code**

Visual Studio Code (often abbreviated as VS Code) is a free, open-source code editor developed by Microsoft for Windows, Linux, and macOS operating systems. It is widely used by developers and programmers for developing and debugging code for various programming languages such as Python, JavaScript, and C++.

VS Code provides a user-friendly interface with features like syntax highlighting, auto completion, code formatting, debugging, and Git integration. It also supports various extensions, which can be installed to add new functionalities, such as additional language support, themes, and snippets.

One of the main advantages of VS Code is its ability to work with a wide range of programming languages and frameworks, making it a popular choice for developers working on different projects. It is also lightweight and fast, with a low memory footprint, making it easy to use even on older or lower-end computers. Overall, VS Code is a versatile and powerful code editor that is widely used by developers across various industries and programming languages.

- **Postman**

One of the most widely used software testing tools for API testing is Postman. Developers maysimply create, test, share, and document APIs with this tool. Postman is a stand- alone API (Application Programming Interface) testing platform for creating, testing, designing, modifying, and documenting APIs. It's a straightforward GUI for sending and viewing HTTP requests and answers.

For testing purposes, Postman eliminates the requirement to develop HTTP client network code.

Instead, we create groups of tests and let Postman interact with the API.

CHAPTER 3

SYSTEM ANALYSIS

3.1 FEASIBILITY STUDY

Conduct research to understand the demand for task management systems in the target market. Identify potential competitors and analyze their offerings, strengths, and weaknesses. Determine if there is a gap in the market that your system can address effectively.Identify the specific needs and requirements of potential users for the task management system. Conduct surveys, interviews, or focus groups with target users to gather insights into their workflow, pain points, and desired features.

Assess the technical feasibility of developing the task management system. Evaluate the availability of technologies, tools, and frameworks required for development. Consider factors such as scalability, performance, compatibility with existing systems, and integration with third-party services.Determine the resources required for developing and maintaining the task management system. This includes human resources (developers, designers, testers), infrastructure (hardware, software, hosting), and financial resources (budget, funding).Estimate the costs associated with developing, deploying, and maintaining the task management system. Compare these costs with the potential benefits, such as increased productivity, efficiency gains, cost savings,and revenue generation. Determine if the expected benefits outweigh the costs.Financial feasibility assessesthe costs associated with developing, implementing, and maintaining the task management system compared to the potential benefits it would bring. It involves estimating initial development costs, ongoing operational expenses, potential savings or revenue generated, and the return on investment (ROI) over time.

Operational feasibility examines whether the proposed task management system aligns with the organization's operations, processes, and goals. It considers factors such as user acceptance, ease of use, integration with existing workflows, and potential disruptions during implementation. Stakeholder buy-in and support are critical considerations in this aspect.This aspect evaluates whether the proposed system complies with relevant laws, regulations, industry

standards, and organizational policies. It includes considerations such as data privacy, security requirements, intellectual property rights, and any legal or regulatory implications associated with the system's development and use.

Schedule feasibility assesses the timeline for developing and implementing the task management system. It involves estimating the time required for each phase of the project, identifying potential bottlenecks or delays, and ensuring that the proposed schedule aligns with organizational timelines and priorities.A thorough risk analysis identifies potential risks and challenges that could affect the success of the task management system project. It involves identifying and assessing risks related to technology, resources, stakeholders, external factors, and other relevant aspects. Strategies for mitigating or managing these risks should also be proposed.

As part of the feasibility study, it's important to explore alternative solutions to task management, such as using existing software, outsourcing development, or adopting different methodologies. Comparing these alternatives helps in making informed decisions about the most suitable approach.Based on the findings of the feasibility study, recommendations are made regarding whether to proceed with the development and implementation of the task management system. These recommendations should consider the overall feasibility, potential benefits, risks, costs, and alignment with organizational objectives.

- Economic feasibility

  The construction of this app is inexpensive. The organization did not have to spend a lot of money to set up a similar system that already existed. What remains to be done is to createan environment for development that is closely monitored. If we achieve this, we may beable to increase the usability of connected services. Even after development, theorganization will not be able to invest extra money in it. As a result, the system is profitable.

- Technical feasibility

  We can say with certainty that it is technically possible as gathering the necessary

resources for system development and maintenance will be easier. All the resources needed to develop and maintain the system are accessible to the company, and we use them.

- Probability feasibility

We can state with certainty The availability of a product refers to its capacity to function. Some items may perform admirably throughout creation and usage, yet they may fail in use.It necessitates doing background checks on the necessary persons as well as testing theirtechnical expertise. Data, updated information, and reports for generations are accurate andtimely in the current system.

3.2 EXISTING SYSTEM

By examining existing systems, you can identify any shortcomings or gaps that users currently experience. This information helps in designing a task management system that addresses these deficiencies effectively.Analyzing existing systems allows you to compare their features, functionalities, and user experiences with what you plan to offer. This comparison can help you identify unique selling points or features that differentiate your system from competitors.

Understanding how existing systems are used within an organization provides insights into potential integration points. Your task management system may need to integrate with other tools or platforms already in use, such as email clients, calendars, project management software, or customer relationship management (CRM) systems.These are basic task management tools that allow users to create, organize, and prioritize tasks in a list format. Examples include Todoist, Microsoft To Do, and Any.do. These apps often include features like due dates, reminders, and task categorization.

Gather feedback from users of existing systems to understand their pain points, preferences, and requirements. This feedback can inform the design and development of your task management system, ensuring that it meets user needs and expectations. Project management tools like Asana, Trello, and Basecamp offer more robust task management

capabilities designed for team collaboration on larger projects. They allow users to create tasks, assign them to team members, track progress, set deadlines, and communicate within the platform.

Assess the underlying technologies, architectures, and infrastructures of existing systems to understand any technical constraints or requirements. This information helps in planning the development and deployment of your task management system, ensuring compatibility and interoperability with existing systems.Time tracking tools like Toggl and Harvest often include task management features that allow users to associate tasks with specific time entries. This can help track time spent on different tasks and projects for billing purposes or performance analysis.

ERP systems like SAP and Oracle often include task management modules as part of their broader suite of business management tools. These systems integrate task management with other functions such as finance, human resources, and supply chain management.Issue tracking tools like Jira and GitHub Issues are commonly used in software development for managing bugs, feature requests, and other tasks related to software projects.

Workflow automation tools like Zapier and Microsoft Power Automate enable users to automate repetitive tasks and workflows across various applications. They often include task management features to create and manage automated tasks.Many organizations develop custom task management systems tailored to their specific needs and workflows. These systems are often built in-house or by third-party developers and can range from simple, lightweight solutions to complex, enterprise-grade platforms.

3.3 DRAWBACK OF EXISTING SYSTEM

The drawbacks for existing system are:

- Task management systems can sometimes be overly complex, with numerous features and options that may overwhelm users, leading to confusion and reduced productivity.
- If the user interface is poorly designed or unintuitive, users may struggle to navigate the system efficiently, resulting in frustration and resistance to adoption.
- Task management systems often need to integrate with other tools and platforms such as calendars, email clients, or project management software. If integration capabilities are limited, it can lead to manual data entry, duplication of efforts, and inefficiencies.
- With the increasing reliance on mobile devices for work, a task management system that lacks adequate mobile support can be a significant drawback. Users may need to access and manage tasks on the go, and if the mobile experience is subpar, it can hinder productivity.
- Every team has unique workflows and preferences, so a task management system that lacks customization options may not fully meet the specific needs of its users. This can lead to workarounds and inefficiencies as teams try to adapt to the system's limitations.
- Effective task prioritization is essential for managing workload and meeting deadlines. If the existing system lacks features or tools to help users prioritize tasks effectively, it can lead to missed deadlines and a sense of overwhelm.
- Slow performance or system crashes can severely disrupt workflow and productivity. If the existing task management system suffers from performance issues, it can lead to frustration among users and hinder their ability to get work done efficiently.
- Without robust analytics and reporting features, it can be challenging for teams to track progress, identify bottlenecks, and make data-driven decisions to improve efficiency and effectiveness.
- Task management systems often contain sensitive information about projects, deadlines, and team members. If the existing system has security vulnerabilities or lacks proper access controls, it can pose a risk to the confidentiality and integrity of this information.

3.4 PROPOSED SYSTEM

Prioritize a user-friendly interface with intuitive navigation and visually appealing design. Conduct user research and usability testing to ensure the system meets the needs and preferences of the target users.Implement real-time collaboration tools such as commenting, task assignment, file sharing, and team messaging to facilitate seamless communication and teamwork among users.Ensure the system supports integration with popular tools and platforms such as calendars, email clients, project management software, and communication tools. Provide APIs and plugins for easy customization and integration with third-party applications.

- Develop mobile applications or responsive web interfaces that offer a seamless task management experience on various devices, allowing users to access and manage tasks anytime, anywhere.A user-friendly interface with an intuitive design is crucial for ease of use and adoption. The proposed system could feature a clean layout, customizable views, and responsive design to accommodate different devices and screen sizes.
- Provide flexible customization options to adapt the system to the unique workflows and preferences of different teams and users. Allow users to customize task views, fields, labels, and notifications according to their needs.Users should be able to easily create tasks, assign them to team members, and set deadlines. The system could include options for categorizing tasks, attaching relevant files or links, and specifying task dependencies.
- Implement features and tools to help users prioritize tasks effectively, such as task dependencies, deadlines, urgency levels, and automated reminders. Provide visualizations and filters to identify high-priority tasks and allocate resources accordingly.Users should have the ability to prioritize tasks based on urgency or importance and track their progress in real-time. Visual indicators, such as color- coded tags or progress bars, can help users quickly identify high-priority tasks or tasks that are overdue.
- The proposed system could facilitate collaboration among team members through features such as task commenting, mentions, and notifications. Integrated communication tools, such as chat or video conferencing, could further streamline collaboration on tasks and projects.Integration with calendar

applications like Google Calendar or Outlook would allow users to view tasks alongside their appointments and meetings. Users could also receive reminders and notifications for upcoming deadlines or scheduled tasks.

- Automation features could help streamline repetitive tasks and workflows, reducing manual effort and improving efficiency. Users could create custom workflows, automate task assignments, or set up recurring tasks.The proposed system could include analytics and reporting tools to provide insights into task completion rates, team performance, and productivity metrics. Customizable dashboards and reports would allow users to track progress and identify areas for improvement.
- Robust security measures, such as user authentication, encryption, and access controls, would be essential to protect sensitive data and ensure compliance with data privacy regulations. The system could also support role-based access control to restrict access to sensitive information based on user roles and permissions.

3.5 BENEFITS OF PROPOSED SYSTEM:

- The user-centric design and intuitive interface of the proposed system enhance usability, making it easier for users to navigate, organize, and prioritize tasks effectively. This leads to increased user adoption and satisfaction.
- Robust collaboration features such as real-time editing, commenting, and file sharing facilitate seamless communication and teamwork among users. This fosters collaboration, reduces silos, and improves overall productivity.
- Integration capabilities enable seamless connectivity with other tools and platforms, eliminating the need for manual data entry and reducing duplication of efforts. This streamlines workflows, saves time, and boosts efficiency.
- A task management system often provides transparency into the status of tasks and projects. Team members can see who is responsible for each task, its current status, and any updates or comments related to it. This transparency helps in accountability and fosters trust within the team.
- Task management systems typically include features for setting deadlines and reminders. This helps

individuals and teams stay on track and ensures that important deadlines are not missed.

- Depending on the system, users may have the ability to customize workflows, task categories, priority levels, and other aspects to fit their specific needs and preferences.
- Some task management systems offer analytics and reporting features that provide insights into team performance, task completion rates, bottlenecks, and other metrics. This data can be valuable for identifying areas for improvement and making informed decisions.
- Many task management systems integrate with other productivity tools such as calendars, email clients, project management software, and communication platforms. This streamlines workflows and reduces the need to switch between different applications.
- One of the most significant task management advantages is that it dramatically increases your productivity when used correctly. It removes the need to balance time taking tasks and allows you to easily prioritize your task list for increased task completion.By creating a visualized priority list, you can complete the most critical tasks first and increase the productivity of your entire team.
- With productivity growth, efficiency will skyrocket as well. More efficiency will bring better opportunities for people who have a hard time being efficient in completing their tasks. This also creates opportunities to analyze the dead zones in your workflow and optimize them for maximum efficiency.
- The other more positive side of task management is its close link with lowering stress and workload. Many people use task management apps to reduce the last-minute stress of tasks. Task management was introduced to save people from their hectic and stressful routines and provide them with enough ease in their daily lives.

3.6 EXISTING SYSTEM

The scope of a project in developing a task management system outlines the boundaries, objectives, deliverables, and functionalities of the system. Here's a breakdown of the scope for a task management system project:

- Clearly define the main objective of the project, which is to develop a comprehensive task management system to streamline task organization, assignment, tracking, and collaboration within an organization or team.

- Identify the core features and functionalities that the task management system will include. This could include task creation, assignment, prioritization, progress tracking, collaboration tools, notifications, reporting, and integration with other tools or systems.
- Define different user roles (e.g., administrators, managers, team members) and their respective permissions within the system. Determine what actions each role can perform, such as creating tasks, assigning tasks, viewing reports, and managing user accounts.
- Specify the platforms and devices the task management system will support, such as web browsers, mobile devices (iOS, Android), and desktop applications. Consider whether the system will be accessible via a web application, mobile app, or both.
- Identify any third-party tools, applications, or systems that the task management system will need to integrate with, such as email clients, calendar applications, project management software, or collaboration platforms. Determine the scope of integration and data exchange between the task management system and other systems.
- Outline how data will be managed, stored, and secured within the task management system. Define data security measures, encryption protocols, user authentication mechanisms, and access controls to protect sensitive information and ensure compliance with data privacy regulations.
- Define the user interface (UI) design principles, navigation structure, layout, and visual elements of the task management system. Ensure that the UI is intuitive, user-friendly, and accessible to users of varying technical proficiency.
- Specify the testing methodologies, tools, and processes that will be used to ensure the quality, reliability, and performance of the task management system. This may include unit testing, integration testing, user acceptance testing (UAT), and performance testing.
- Develop a deployment plan outlining the steps and timeline for deploying the task management system to users. Consider factors such as rollout strategy, training requirements, user onboarding, and post-deployment support.
- Create a project timeline with key milestones and deliverables, including design phase, development sprints, testing phases, deployment, and ongoing maintenance. Establish realistic .

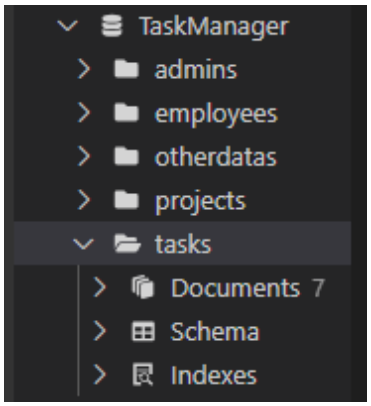
CHAPTER 4

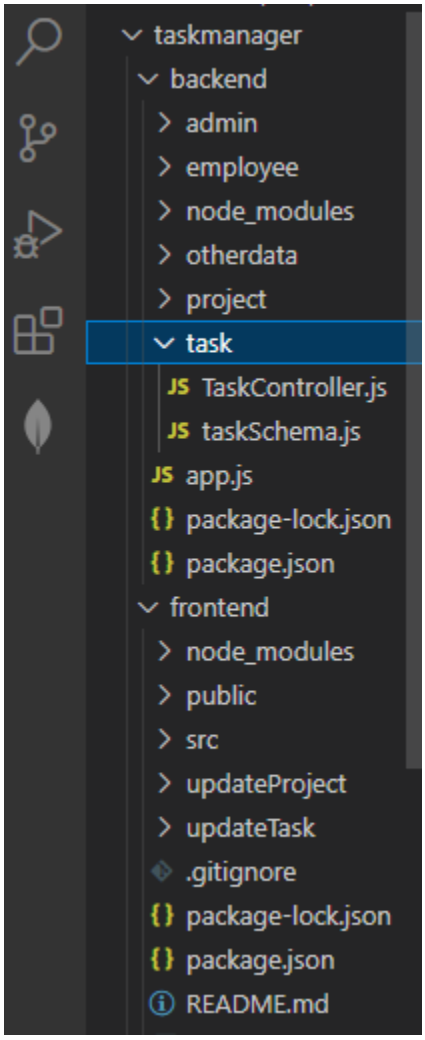
SYSTEM DESIGN

4.1 DATA FLOW DIAGRAM

4.2 ARCHITECTURE DIAGRAM

4.3 DATABASE DESIGN




taskmanager
backend
admin
employee
node_modules
otherdata
project
task
TaskController.js
taskSchema.js
app.js
package-lock.json
package.json
frontend
node_modules
public
src
updateProject
updateTask
.gitignore
package-lock.json
package.json
README.md

CHAPTER 5

PROJECT DESCRIPTION

5.1 OBJECTIVE

The main objective of the project are as follows:

- The Task Manager acts as a central hub for collaboration, task tracking, responsibility assignment, and deadline monitoring, enhancing project efficiency and organization.
- By utilizing the Task Manager, project stakeholders gain real-time visibility into project status, enabling them to identify bottlenecks, proactively address challenges and make informed decisions to keep the project on track towards achieving its objectives.

5.2 MODULE DESCRIPTION

This project consists of mainly three modules and they are listed below:

- Admin / Employee Login
- Task / Project Assignment
- Adding Employees
- View Task / Project
- Update Task status
- Edit profile

5.2.1 Admin/ Employee Login:

- Implemented secure password hashing algorithms like bcrypt to store and validate passwords securely.Passwords are securely hashed using bcrypt for storage and validation.
- A Forgot Password feature has been implemented, allowing users to reset their passwords securely through email verification.
- Rate-limiting and account lockout mechanisms prevent brute force attacks, enhancing

security.Multi-factor authentication (MFA) adds an extra layer of security to the login process.

- Login attempts and security-related events are logged for auditing purposes.The login screens are customized with branding and styling options for a cohesive user experience.
- Users have the option to remember their login sessions for convenience.Session management handles session expiration, renewal, and revocation securely.
- HTTPS and SSL/TLS encryption ensure that communication between clients and servers during login is secure.

5.2.2 Task/ Project Assignment:

- Admin can create tasks/projects easily with an intuitive interface and comprehensive form fields.
- Adding / creating task or project button will be available in Admin dashboard.
- Admins can assign tasks/projects to multiple employees simultaneously, streamlining workflow management.
- Task dependencies have been implemented, ensuring that tasks cannot start until prerequisite tasks are completed.
- Files or documents can be attached to tasks/projects, providing additional context and resources.
- Recurring tasks/projects are supported for routine activities, reducing manual effort.
- Tasks/projects can be prioritized based on urgency or importance, aiding in effective resource allocation.
- Task/project templates are available for common workflows or processes, saving time and ensuring consistency.
- Tasks/projects can be assigned based on employee workload and availability, optimizing resource utilization.

5.2.3 Adding Employees:

- New employee registration includes validation checks to ensure completeness and accuracy of information.
- Email verification is required for new employee accounts, enhancing security and preventing

unauthorized access.

- Admins can categorize employees into departments or teams for better organization and management.
- An option to invite employees via email with a unique registration link has been implemented for seamless onboarding.
- Role-based permissions restrict certain actions or access levels for different employee roles, ensuring data security.
- Employee data can be imported from external sources such as CSV files or HR systems, simplifying the onboarding process.
- An intuitive onboarding process guides new employees through profile completion and preference settings.
- User-friendly error messages and feedback are provided during employee registration to assist users.

5.2.4 View Task/ Project:

- Tasks/projects can be sorted and filtered based on various attributes such as status, priority, and due date, enabling efficient task management.
- Customizable views/layouts (e.g., list view, tables) cater to users' preferences and workflow requirements.
- Search functionality allows users to quickly locate specific tasks/projects using keywords or criteria.
- Real-time updates or notifications keep users informed of changes to task/project statuses or details.Tagging or labeling systems categorize tasks/projects for easy organization and retrieval.
- Task/project histories are accessible, allowing users to track changes and monitor progress over time.

5.2.5 Update Task Status:

- An intuitive interface with drag-and-drop functionality facilitates easy updating of task/project

statuses.

- Automated notifications ensure that relevant stakeholders are informed of status changes in real-time.Users can leave comments or notes when updating task/project statuses, providing context and updates.
- A centralized dashboard or summary view provides an overview of overall progress and status across all tasks/projects.
- Role-based permissions restrict certain users from modifying task/project statuses, ensuring data integrity.
- Bulk status updates streamline workflow management by allowing users to update multiple tasks/projects simultaneously.
- Reminders or alerts for overdue tasks/projects ensure timely completion and adherence to deadlines.
- Custom status labels or categories can be set based on users' workflow requirements.

5.2.6 Edit Profile:

- Reminders or alerts for overdue tasks/projects ensure timely completion and adherence to deadlines.
- Users can upload and manage their profile pictures or avatars, personalizing their accounts.Contact information such as phone number, address, and social media profiles can be updated easily.
- Localization options support multiple languages or regional preferences for profile settings.Dark mode or theme customization options enhance user experience and accessibility.
- Users can customize notification preferences for task/project updates, reminders, or mentions.
- An activity log tracks profile changes and account activities for auditing and accountability.
- External account linking (e.g., Google, Microsoft) enables single sign-on or integration with other platforms.Privacy settings allow users to control the visibility of their profile information and activity.

5.3 Technologies Used:

- MongoDB: Document-oriented NoSQL database for data storage.
- Express.js: Web application framework for server-side routing and middleware.
- ReactJS: Front-end JavaScript framework for interactive user interfaces and MVC architecture.
- Node.js: Server-side JavaScript runtime for event-driven and asynchronous task execution.

5.4 Benefits:

- Improved productivity and efficiency in task management.
- Enhanced collaboration and communication within project teams.
- Better visibility into task status, progress, and timelines.
- Streamlined workflow and optimized resource allocation.
- Data-driven insights for informed decision-making and performance analysis.

5.5 Implementation:

STEP 1:

Run both frontend and backend code sing “npm start”.

STEP 2:

Once the frontend started login as an admin or employee to start the application.

STEP 3:

After logged in successfully, a dashboard will be displayed, there all data about the Application will be displayed.

STEP 4:

Create Task/ project as an admin or create a profile for employee. The data will be stored in database respectively.

STEP 5:

Click on “view Task” or “view project” or “view employee”, the corresponding data will be displayed in table format.

STEP 6:

Update any data as an admin, so the changes will reflected in database.

STEP 7:

Run the application as an employee, the corresponding person’s profile, assigned task and projects will be displayed. Update the status of the assigned task to make the permanent change in database.

CHAPTER 6

SYSTEM TESTING

6.1 TESTING DEFINITION

Testing is performed to identify errors. It is used for quality assurance. Testing is an integral part of the entire development and maintenance process. The goal of the testing during phase is to verify that the specification has been accurately and completely incorporated into the design, as well as to ensure the correctness of the design itself. For example, the design must not have any logic faults in the design is detected before coding commences, otherwise the cost of fixing the faults will be considerably higher as reflected. Detection of design faults can be achieved by means of inspection as well as walkthrough.

6.2 TESTING OBJECTIVE

Testing is one of the important steps in the software development phase. Testing checks for the errors, as a whole of the project testing involves the following test cases:

- Static analysis is used to investigate the structural properties of the Source code.
- Dynamic testing is used to investigate the behavior of the source code by executing the program on the test data.

6.3 TYPES OF TESTING

The types of testing are:

- Unit testing
- Integration testing
- Functional testing
- Performance testing
- User acceptance testing

6.3.1 Unit Testing

This type of testing focuses on testing individual components or functions of the system, such as the database model or thefrontend module. Unit testing helps to ensure that each component of the system is working correctly and can be integrated into the larger system.

6.3.2 Integration testing

This type of testing focuses on testing how different components of the system work together, such as the integration between the backend model and the frontend module. Integration testing helps to ensure that the system is working as a cohesive unit and that there are no issues with communication or data flow between different components.

6.3.3 Functional testing

This type of testing focuses on testing the functional requirements of the system, such as whether the system correctly detects the data is storing in database through API's. Functional testing helps to ensure that the system is meeting its intended purpose and that all features are working correctly.

6.3.4 Performance testing

This type of testing focuses on testing the performance of the system under different conditions, such as varying lighting conditions or different types of distractions. Performance testing helps to ensure that the system is functioning reliably and accurately under real-world conditions.

6.3.5 User acceptance testing

This type of testing focuses on testing the system from the perspective of the end-users, such as employees or administrators. User acceptance testing helps to ensure that the system is easy to use, intuitive, and meets the needs of the users.

CHAPTER 7

CONCLUSION

7.1 SUMMARY

In conclusion, the Task Manager module within the MERN stack encapsulates a sophisticated and versatile set of tools and functionalities essential for modern project management. Through the seamless integration of MongoDB for robust data storage, Express.js for efficient server-side routing and middleware management, ReactJS for dynamic front-end development using the MVC architecture, and Node.js for scalable server-side execution, this module emerges as a powerhouse for organizations seeking to optimize their task management workflows.

The Task Manager module's key features, including real-time task tracking, agile task assignment mechanisms, intelligent task prioritization algorithms, and intuitive task scheduling capabilities, collectively empower project managers and team leaders to orchestrate complex projects with precision and agility. Moreover, the module's collaboration tools, such as task- related discussions, notifications, and commenting systems, foster enhanced communication and collaboration among team members, leading to increased productivity and synergy within project teams.

Targeting a diverse audience including project managers, team leaders, and developers across various industries and project domains, the Task Manager module delivers tangible benefits such as improved productivity, streamlined workflows, enhanced communication, optimized resource allocation, and informed decision-making. Whether applied in agile project management methodologies, software development projects, or task-based project tracking scenarios, the Task Manager module stands as a cornerstone of efficiency and effectiveness within the MERN stack environment, driving projects towards successful outcomes and organizational growth.

7.2 FUTURE ENHANCEMENTS

In envisioning future enhancements for the Task Manager module within the MERN stack, the focus lies on creating a harmonious blend of simplicity and sophistication. This entails introducing intuitive dashboards that offer insightful project metrics, seamlessly integrating collaboration tools for streamlined communication, and implementing resource management features to optimize resource allocation. Additionally, enhancing the mobile experience with responsive design and robust security measures ensures a cohesive and secure workflow. Integrating user feedback mechanisms completes the symphony, enabling continuous improvement and ensuring that the Task Manager module evolves in tune with the dynamic needs of project management in today's digital landscape.

CHAPTER 8

APPENDIX

8.1 SCREENSHOTS

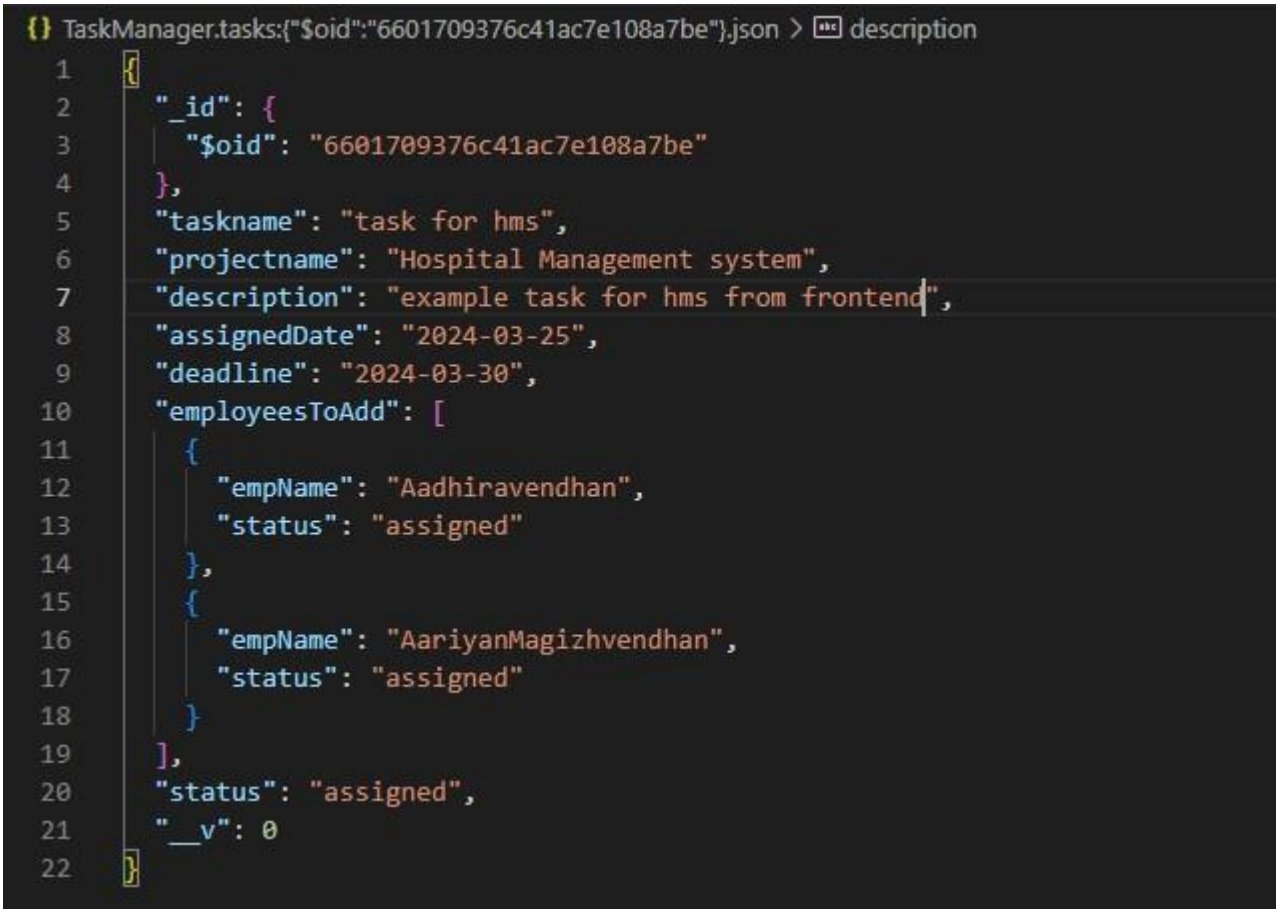


Fig 8.1.1 Data storing format in MongoDB

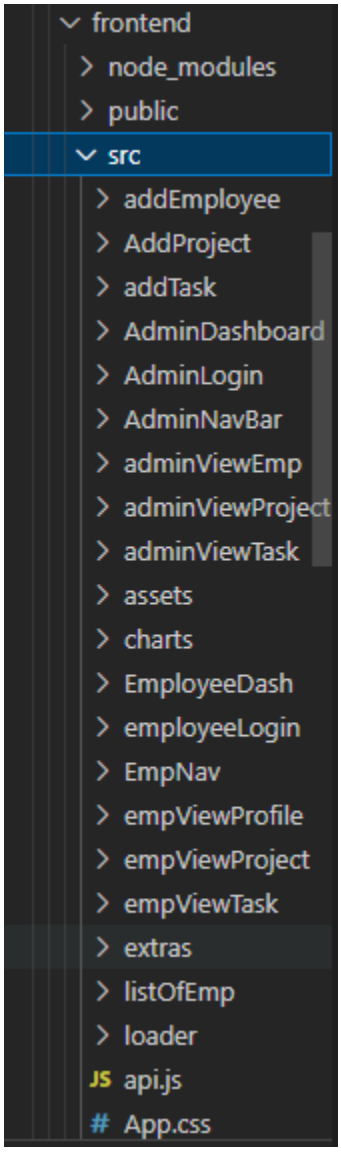


Fig 8.1.2 Folder structure of Frontend

Fig 8.1.3 Home page of the application

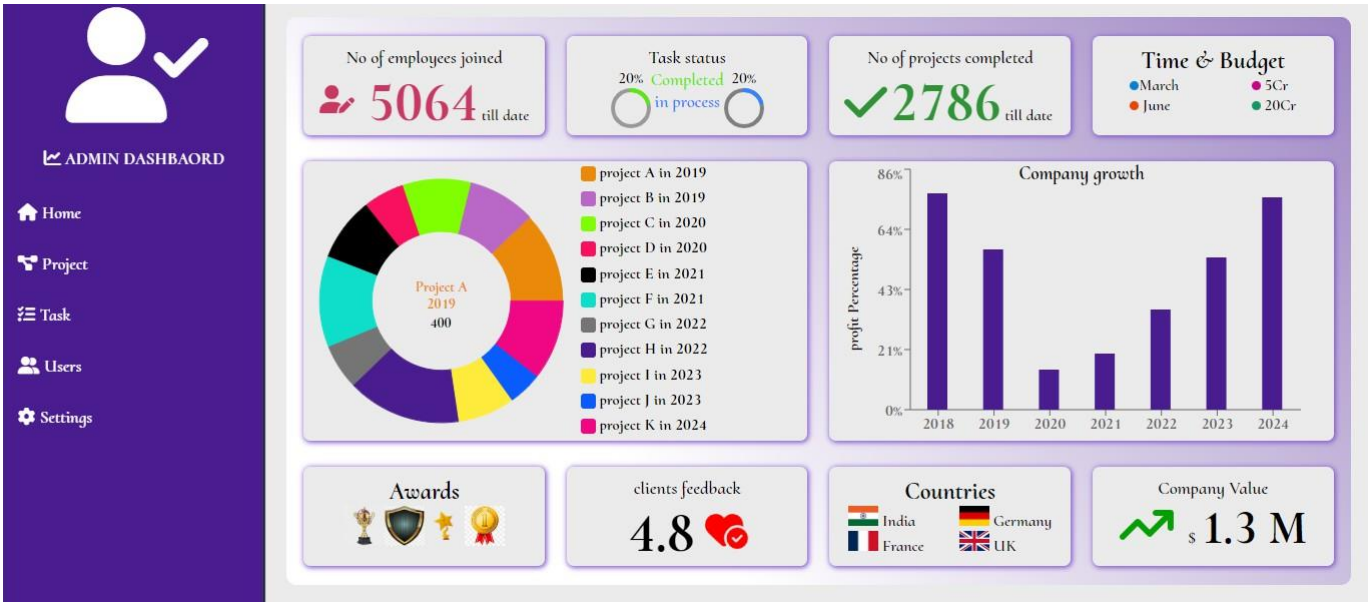


Fig 8.1.4 Admin dashboard

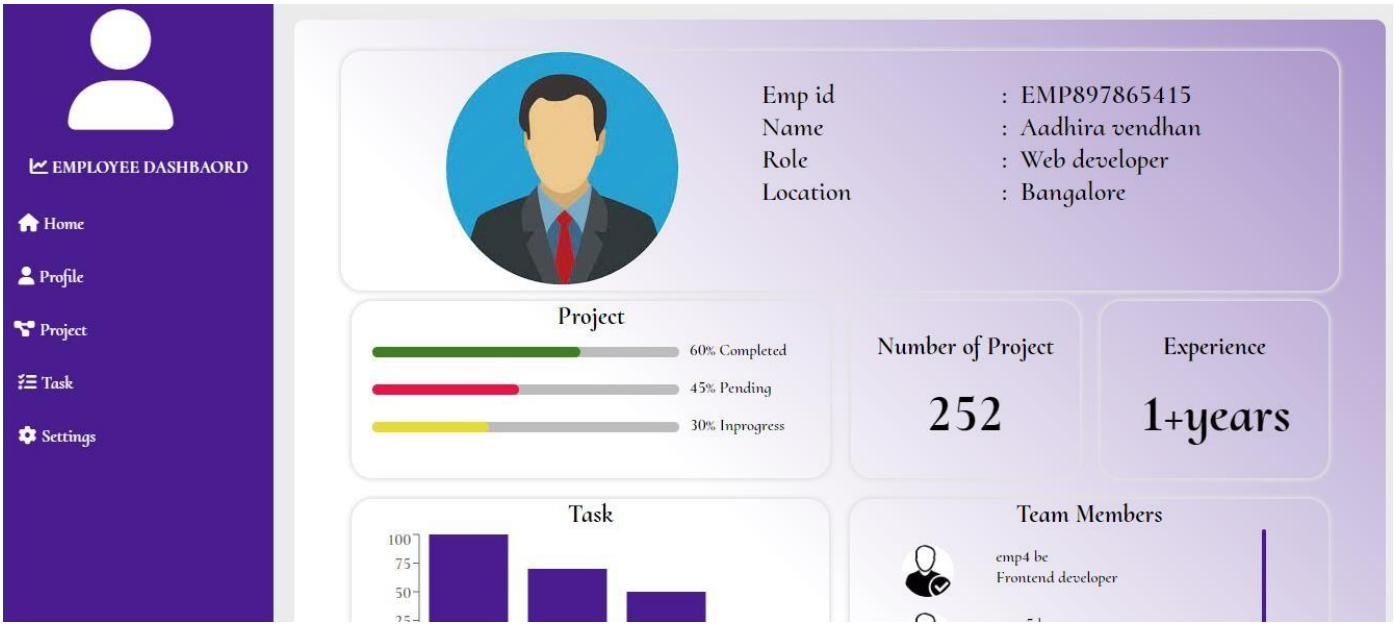


Fig 8.1.5 Employee dashboard

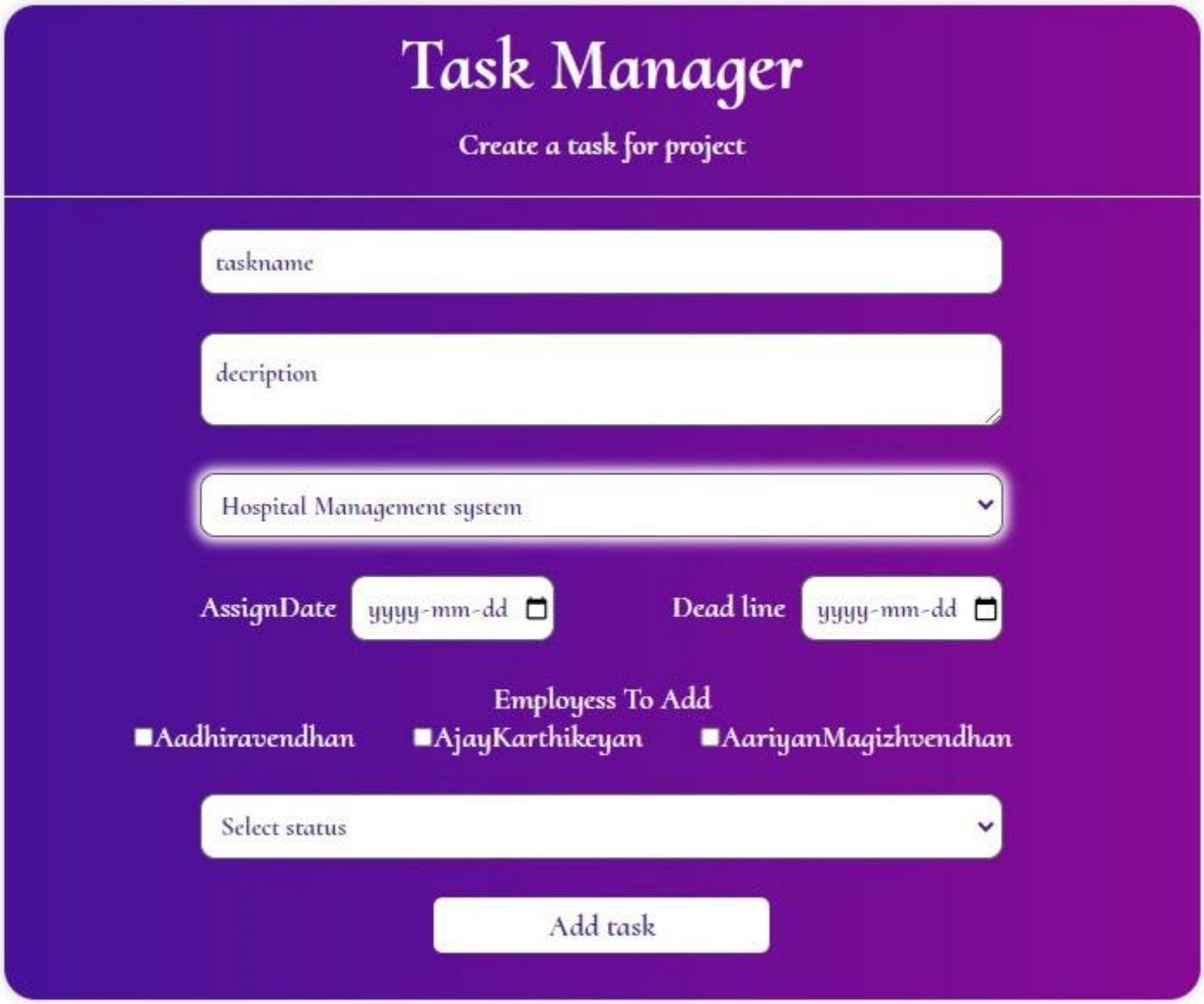


Fig 8.1.6 Assigning task

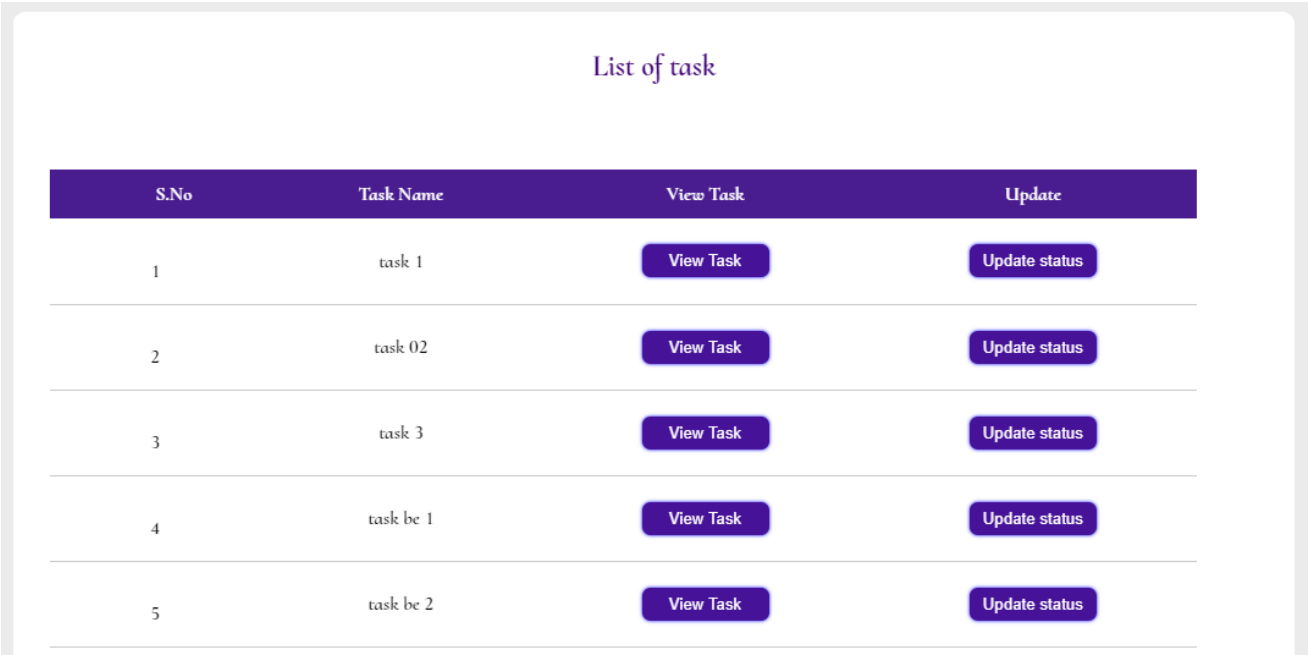


Fig 8.1.7 Displaying all tasks

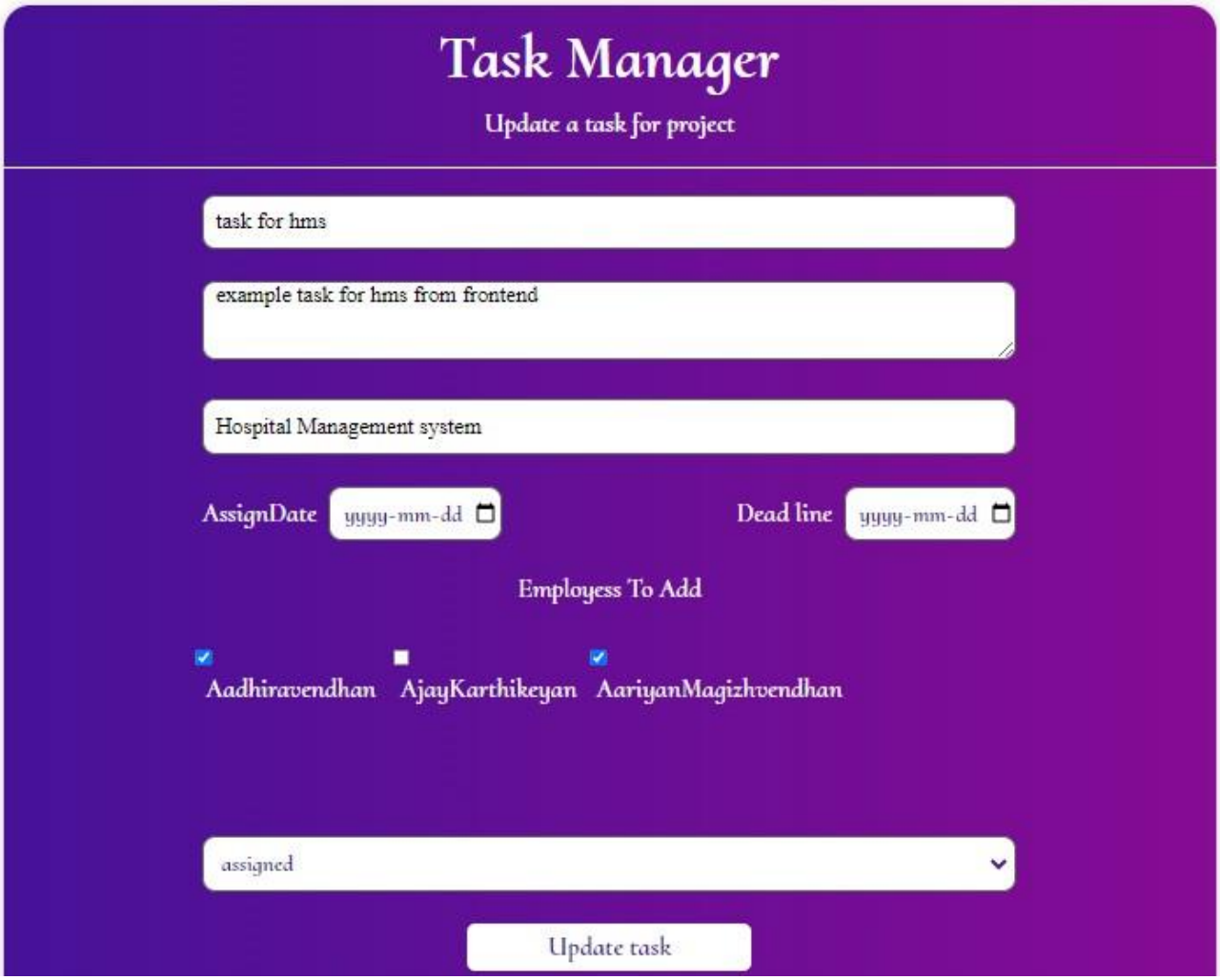


Fig 8.1.8Updating the task

Fig 8.1.9 Profile of an employee

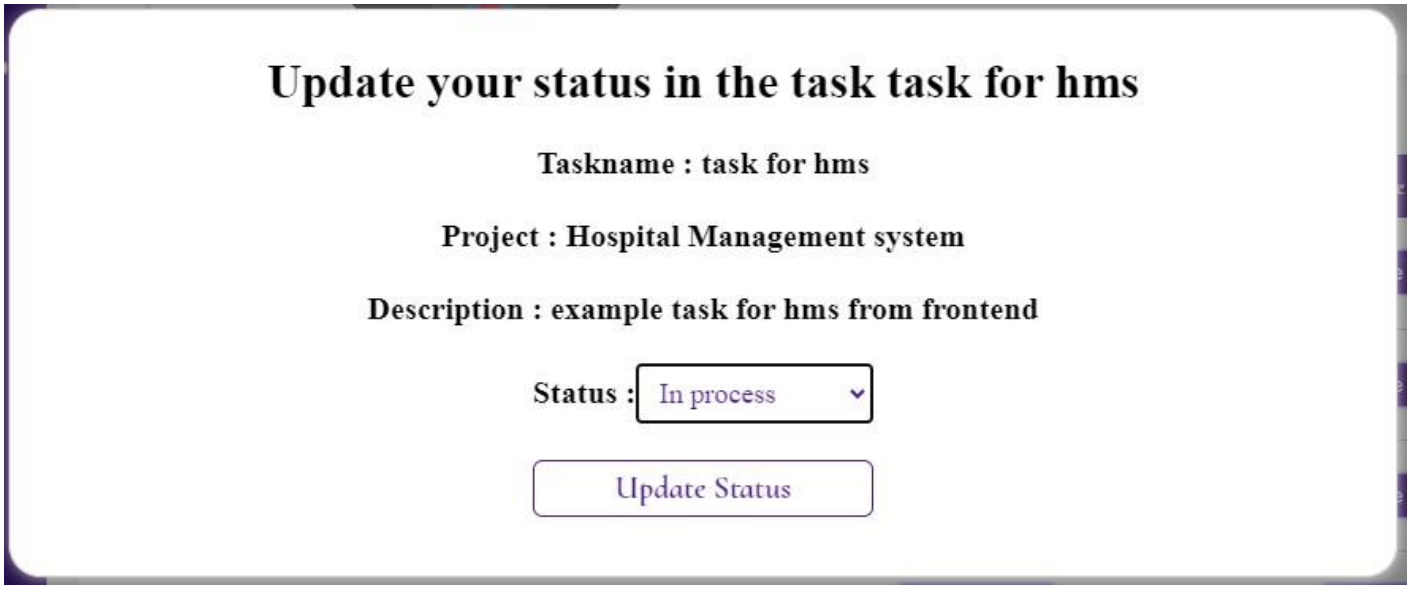


Fig 8.1.10 Updating status of a task by employee

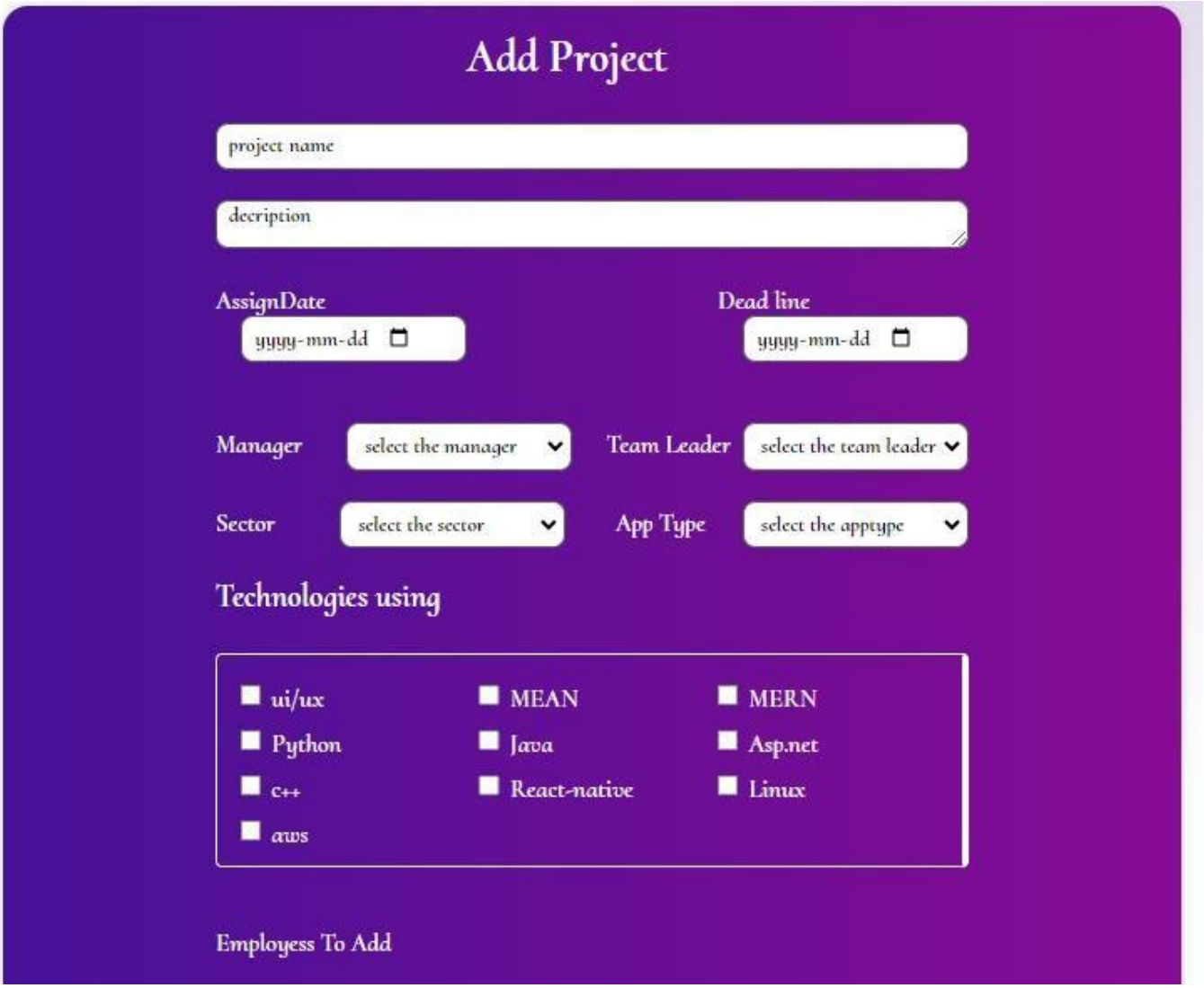


Fig 8.1.11(a) Adding project by Admin

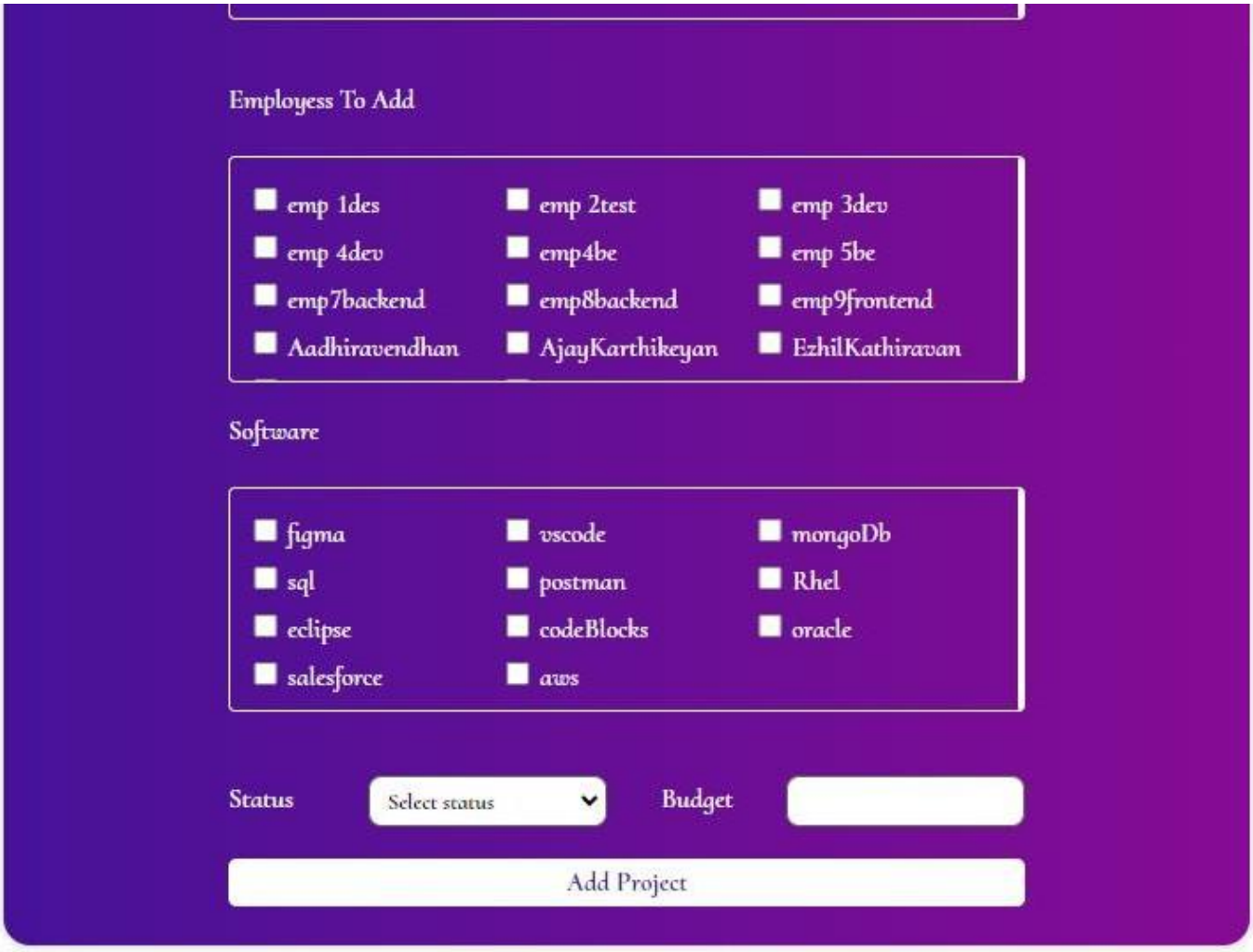


Fig 8.1.11(b) Adding project by Admin

8.2 CODING

8.1 ADMIN LOGIN (jsx)

```
import { useState} from 'react';
import { useNavigate } from 'react-router-dom';
import axios from 'axios';

import './AdminLogin.css';

function AdminLogin(){
   const [loginData, setLoginData] = useState({
username : "",
password : ""
    });
    const [err1, setErr1] = useState({});
    const [err2, setErr2] = useState("")
    const navigate = useNavigate()

    const changeLogin = (e) => {
setLoginData({...loginData, [e.target.name] : e.target.value });
    }

    const validateAdminLogin = (loginData) =>{
      let err1 = {}
if(loginData.username === ""){
          err1.ErrLogin = "Please enter username"
      }
```

```
    else if(loginData.password === ""){
        err1.ErrLogin = "Please enter password"
    }

    return err1;
  }

  const submitLogin = (e) => {
e.preventDefault();
    console.log(loginData);
    const errors = validateAdminLogin(loginData);
    if (Object.keys(errors).length === 0) {

  axios.post(`http://127.0.0.1:3012/taskmanger/inns/api/admin/loginadmin?username=${loginDat
  a.username}`, JSON.stringify(loginData), {
        headers: {
          "Content-Type": "application/json"
        }
      }).then((res) => {
        console.log(res.data);
        navigate('/adminDashboard');
      }).catch((err) => {
        if (err.response&&err.response.data&&err.response.data.message) {
          setErr1({ backend: err.response.data.message });
        } else {
          setErr1({ backend: "An error occurred. Please try again later." });
        }
      });
    } else {
```

```
            setErr1(errors);
        }
      }

return(
<div className='adminTotal'>
<div id="total">
<div className="container">
<div className="screen">
<div className="screen__content">
<h1>Admin Log<span>in</span></h1>
<form className="login">
<div className="login__field">
<i className="login__icon fas fa-user"></i>
<input type="text" className="login__input" placeholder="User name / Email"  name='username'
    value={loginData.username} onChange={changeLogin}/>
</div>
<div className="login__field">
<i className="login__icon fas fa-lock"></i>
<input type="password" className="login__input" placeholder="Password" name='password'
    value={loginData.password} onChange={changeLogin}/>
</div>
<p>{err1.ErrLogin}</p>
<p>{err1.backend}</p>
<button className="button login__submit" onClick={submitLogin}>
<span className="button__text">Log In Now</span>
<i className="button__icon fas fa-chevron-right"></i>
</button>
</form>
```

```
<div className="social-login">
<h3>log in via</h3>
<div className="social-icons">
<a href="#" className="social-login__icon fab fa-instagram"></a>
<a href="#" className="social-login__icon fab fa-facebook"></a>
<a href="#" className="social-login__icon fab fa-twitter"></a>
</div>
</div>
</div>
<div className="screen__background">
<span className="screen__background__shape screen__background__shape4"></span>
<span className="screen__background__shape screen__background__shape3"></span>

<span className="screen__background__shape screen__background__shape2"></span>
<span className="screen__background__shape screen__background__shape1"></span>
</div>
</div>
</div>


</div>


</div>
   )
}
export default AdminLogin;
```

8.2 ADMIN DASHBOARD (jsx)

```
import React, { useEffect, useState } from "react";
import './AdminDashboard.css';
import Navbar from '../AdminNavBar/Nav'
import Barchart from '../charts/barChart'
import Piechart from "../charts/pieChart";
import axios from 'axios';
import award1 from '../assets/award1.jfif';
import award2 from '../assets/award2.jfif';
import award3 from '../assets/award3.png';
import award4 from '../assets/award4.jfif';
import india from '../assets/india.png';
import france from '../assets/france.png';
import germany from '../assets/germany.png';
import uk from '../assets/uk.png';
import PropTypes from 'prop-types';
import Slider, { SliderThumb } from '@mui/material/Slider';
import { styled } from '@mui/material/styles';
import { Progress, ButtonGroup, Button, Row, Col } from 'rsuite';


import { FontAwesomeIcon } from '@fortawesome/react-fontawesome';
import { faUserPen, faCheck, faHeartCircleCheck, faArrowTrendUp} from '@fortawesome/free-solid-
    svg-icons';
import { useNavigate } from "react-router-dom";
```

```
function AdminDashBoard(){
const[getCountOfEmp , setGetCountOfEmp] = useState(0);
const[getCountOfProject , setGetCountOfProject] = useState(0);
const[getCountOfCompTask , setGetCountOfCompTask] = useState(0);
const[getCountOfPendTask , setGetCountOfPendTask] = useState(0);
    const [percent, setPercent] = React.useState(0);
    const [status, setStatus] = useState(null);
    const navigate = useNavigate();
    const [popupProj, setPopupProj] = useState(false);
    const [viewProj, setViewproj] = useState([]);
    const [projectname, setProjectname] = useState('');
    const [allProj, setAllProj] = useState([]);

useEffect(()=>{
setStatus((percent === 100) ? 'success' : null);

        // count employee
axios.get('http://127.0.0.1:3012/taskmanager/api/employee/getCountOfEmp',{
headers : {
              "Content-Type" : "apllication/json"
           }
        }).then((res)=>{
           console.log(res.data.getCount);
setGetCountOfEmp(res.data.getCount);
        }).catch((err)=>{
           console.log(err)
        })
```

```
        // count project
axios.get('http://127.0.0.1:3012/taskmanager/api/project/getCountOfPro',{
headers : {
              "Content-Type" : "apllication/json"
           }
        }).then((res)=>{
           console.log(res.data.getCount);
setGetCountOfProject(res.data.getCount);
        }).catch((err)=>{
           console.log(err)
        });

        // get count of completed task
axios.get('http://127.0.0.1:3012/taskmanger/inns/api/task/countOfCompSts?status=completed',{
headers : {
              "Content-Type" : "apllication/json"
           }
        }).then((res)=>{
           console.log(res.data.compStatus);
setGetCountOfCompTask(res.data.compStatus);
        }).catch((err)=>{
           console.log(err)
        });
        // get count of in process task
axios.get('http://127.0.0.1:3012/taskmanger/inns/api/task/countOfCompSts?status=in process',{
headers : {
              "Content-Type" : "apllication/json"
           }
```

```
        }).then((res)=>{
            // console.log(res.data.compStatus);
setGetCountOfPendTask(res.data.compStatus);
        }).catch((err)=>{
            console.log(err)
        });

axios.get('http://127.0.0.1:3012/taskmanger/inns/api/project/viewAllProject', {
            headers: {
              "Content-Type": "application/json"
            }
          }).then((res) => {
            console.log(res.data.viewProject)

            let projects = res.data.viewProject;
            let projectNames = projects.map(project =>project.projectname);
setAllProj(projectNames);
          }).catch((err) => {
            console.log(err)
          })

    }, []);

useEffect(()=>{

},[viewProj])

    const popupProject = async(projectname) => {
        try {
```

```
        const data = await
    axios.get(`http://127.0.0.1:3012/taskmanger/inns/api/project/viewOneProject?projectname=${projec
    tname}`,{
headers : {
          "Content-Type" : "application/json"
        }
       })
setViewproj(data.data.viewProject[0]);
      } catch (error) {
console.error('Error project getting', error);
      }
      (popupProj === false) ?setPopupProj(true) : setPopupProj(false);
  }

  const closePopupProj = () => {
setPopupProj(false)
  }

return(
<div className="totalDashBoardDivadminDashbaord">
<div className="navDiv" id='navDiv'>
<Navbar />
</div>
<div className="content-wrapper admn-dshbrd-wraper" id='content-wrapper'>
<div className="content content-boxes">
<div className={(popupProj == true) ? 'active' : 'deactive'}>
<div className="contentActive">
<b id="closeActive" onClick={closePopupProj}>&times;</b>
<h1>{viewProj.projectname}</h1>
```

```
<div>
<p><span>description :</span>{viewProj.description}</p>
<p><span>assignedDate :</span> {viewProj.assignedDate}</p>
<p><span>deadline : </span>{viewProj.deadline}</p>
<p><span>applicationType :</span>{viewProj.applicationType}</p>
<p><span>sector :</span> {viewProj.sector}</p>
<p><span>status :</span> {viewProj.status}</p>
<p><span>budget :</span> {viewProj.budget}</p>
<p><span>manager :</span>{viewProj.manager}</p>
<p><span>teamLeader :</span>{viewProj.teamLeader}</p>
<p><span>technologies :</span> {viewProj.technologies}</p>
<p><span>tasks : </span>{viewProj.tasks}</p>
<p><span>employeesToAdd :</span> {viewProj.employeesToAdd}</p>
<p><span>softwareUsing :</span> {viewProj.softwareUsing}</p>
</div>
</div>
</div>
<div className='content-box3 content-box-common' id='contbox7' onClick={()=>
    navigate('/ListOfEmp')}>
<p>No of employees joined</p>
<h1><FontAwesomeIcon icon={faUserPen} style={{width:"35px"}}/> {5050+getCountOfEmp}</h1>
    till date
</div>
<div className='content-box4 content-box-common' id='contbox8'>
<p>Task status</p>
<div className="loaderPercent">
<div style={{ width: 40 }}>
<Progress.Circle percent={getCountOfCompTask*10} strokeColor={'#59e90c'} strokeWidth={10}
    status={status} trailColor="#9b9999" trailWidth={10}/>
```

```
</div>
<div>
<p style={{color:'#59e90c'}}>Completed</p>
<p style={{color:'#3385ff'}}>in process</p>
</div>
<div style={{ width: 40 }}>
<Progress.Circle percent={getCountOfPendTask*10} strokeColor={'#3385ff'} strokeWidth={10}
    status={status} trailColor="grey" trailWidth={10}/>
</div>
</div>
</div>
<div className='content-box5 content-box-common' id='contbox9'>
<p>No of projects completed</p>
<h1>
<FontAwesomeIcon icon={faCheck}  id="faCheck"/>
                {2780+getCountOfProject}</h1> till date
</div>
<div className='content-box6 content-box-common' id='contbox10'>
<h1>Time & Budget</h1>
<ul>
<span>
<li><span></span>March</li>
<li><span></span> June</li>
</span>
<span>
<li><span></span> 5Cr</li>
<li><span></span> 20Cr</li>
</span>
</ul>
```

```
</div>
<div className='content-box1 content-box-common' id='contbox1'>
<div className="boxInBox">
<Piechart />
</div>
<div className="boxInBox">
<div onClick={()=>popupProject(allProj[0])}>
<span></span><b>project A in 2019</b>
</div>
<div onClick={()=>popupProject(allProj[1])}>
<span></span><b>project B in 2019</b>
</div>
<div onClick={()=>popupProject(allProj[2])}>
<span></span><b>project C in 2020</b>
</div>
<div onClick={()=>popupProject(allProj[3])}>
<span></span><b>project D in 2020</b>
</div>
<div onClick={()=>popupProject(allProj[4])}>
<span></span><b>project E in 2021</b>
</div>
<div onClick={()=>popupProject(allProj[0])}>
<span></span><b>project F in 2021</b>
</div>
<div onClick={()=>popupProject(allProj[1])}>
<span></span><b>project G in 2022</b>
</div>
<div onClick={()=>popupProject(allProj[2])}>
<span></span><b>project H in 2022</b>
```

```
</div>
<div onClick={()=>popupProject(allProj[3])}>
<span></span><b>project I in 2023</b>
</div>
<div onClick={()=>popupProject(allProj[4])}>
<span></span><b>project J in 2023</b>
</div>
<div onClick={()=>popupProject(allProj[0])}>
<span></span><b>project K in 2024</b>
</div>
</div>
</div>
<div className='content-box2 content-box-common' id='contbox2'>
<div>
<h3>Company growth</h3>
</div>
<div className="percent">
<h4>profit Percentage</h4>
</div>
<Barchart />
</div>
<div className='content-box3 content-box-common' id='contbox3'>
<h1>Awards</h1>
<div>
<span className="awardsCont">
<imgsrc={award1} alt="" width={'40px'} height={'40px'}/>
<span>
                        Best Company in 2020
</span>
```

```
</span>
<span className="awardsCont">
<imgsrc={award2} alt="" width={'40px'} height={'40px'}/>
<span>
                        Best Company in 2018
</span>
</span>
<span className="awardsCont">
<imgsrc={award3} alt="" width={'40px'} height={'40px'}/>
<span>
                        Best Company in 2020
</span>
</span>
<span className="awardsCont">
<imgsrc={award4} alt="" width={'40px'} height={'40px'}/>
<span>
                        Best Company in 2023
</span>
</span>
</div>
</div>
<div className='content-box4 content-box-common' id='contbox4'>
<p>clients feedback</p>
<h1>4.8 <FontAwesomeIcon icon={faHeartCircleCheck} id="heartbeatIcon"/></h1>
</div>
<div className='content-box5 content-box-common' id='contbox5'>
<h1>Countries</h1>
<ul>
<span>
```

```
<li><imgsrc={india} alt="" width={'30px'} height={'20px'}/> India</li>
<li><imgsrc={france} alt="" width={'30px'} height={'20px'}/> France</li>
</span>
<span>
<li><imgsrc={germany} alt="" width={'30px'} height={'20px'}/> Germany</li>
<li><imgsrc={uk} alt="" width={'30px'} height={'20px'}/> UK</li>
</span>
</ul>
</div>
<div className='content-box6 content-box-common' id='contbox6'>
<p>Company Value</p>
<h3><FontAwesomeIcon icon={faArrowTrendUp} id="faCaretUp"/></h3>$ <h3>1.3 M</h3>
</div>
</div>
</div>
</div>
   )
}
export default AdminDashBoard;
```

8.3 ADMIN DASHBOARD.CSS

```
.adminDashbaord{
  font-family: 'Caudex';
}

.adminDashbaord .admn-dshbrd-wraper{
  width: 80.7vw;
  height: 100vh;
}
```

```
.adminDashbaord .content-boxes{
  height: 39.99vw;
  padding: 6px;
  display: grid;
  grid-template-columns: 25% 25% 25% 25%;
  grid-template-rows: 22.5% 55% 22.5%;
  grid-template-areas: 'contbox7 contbox8
     contbox9 contbox10'
                  'contbox1 contbox1 contbox2
     contbox2'
                  'contbox3 contbox4 contbox5
     contbox6';
}

.adminDashbaord .content-boxes #contbox1{
  grid-area: contbox1;
}
.adminDashbaord .content-boxes #contbox2{
  grid-area: contbox2;
}
.adminDashbaord .content-boxes #contbox3{
  grid-area: contbox3;
}
.adminDashbaord .content-boxes #contbox4{
  grid-area: contbox4;
}
.adminDashbaord .content-boxes #contbox5{
  grid-area: contbox5;
```

```
}
```

```
.adminDashbaord .content-boxes #contbox6{
  grid-area: contbox6;
}
.adminDashbaord .content-boxes #contbox7{
  grid-area: contbox7;
  border-bottom: 0px solid #c83863;
  transition: all .1s linear;
}
.adminDashbaord .content-boxes #contbox8{
  grid-area: contbox8;
}
.adminDashbaord .content-boxes #contbox9{
  grid-area: contbox9;
  border-bottom: 0px solid #2fd339;
  transition: all .1s linear;
}
.adminDashbaord .content-boxes #contbox10{
  grid-area: contbox10;
}

.adminDashbaord #contbox7:hover{
  border-bottom: 5px solid #c83863;
}
.adminDashbaord #contbox9:hover{
  border-bottom: 5px solid #2fd339;
}

.adminDashbaord .content-boxes .content-box1{
  display: flex;
```

```
  flex-direction: row;
}

.content-boxes .content-box3,.content-boxes
      .content-box4,
.content-boxes .content-box5,.content-boxes
      .content-box6{
  text-align: center;
  padding-top: 9px;
}

.adminDashbaord .content-boxes .content-box1
      .boxInBox:nth-child(1){
  width: 55%;
}
.adminDashbaord .content-boxes .content-box1
      .boxInBox:nth-child(2){
  width: 45%;
  display: grid;
}

.content-boxes #contbox7 h1,.content-boxes
      #contbox9 h1,
.content-boxes #contbox4 h1,.content-boxes
      #contbox6 h3 {
  display: inline;
  font-size: 3.5em;
}
.content-boxes #contbox9 h1{
```

```
  color: #319337;
}
.content-boxes #contbox7 h1{
  color: #c83863;
}
.content-boxes #contbox9 #faCheck{
  width: 43px;
  position: relative;
  top: 3px;
  right: 4px;

}

.content-boxes #contbox8 .loaderPercent{
  width: inherit;
  display: flex;
  justify-content: center;
}

.content-boxes #contbox5 ul,.content-boxes
    #contbox10 ul{
  list-style-type: none;
  display: flex;
  justify-content: space-around;
  text-align: start;
}
.content-boxes #contbox10 ulspan:nth-child(1)
    li:nth-child(1) span,
.content-boxes #contbox10 ulspan:nth-child(1)
```

```
    li:nth-child(2) span,
.content-boxes #contbox10 ulspan:nth-child(2)
    li:nth-child(1) span,
.content-boxes #contbox10 ulspan:nth-child(2)
    li:nth-child(2) span{
  width: 9px;
  height: 9px;
  display: inline-block;
  border: 1px solid transparent;
  border-radius: 50%;
}
.content-boxes #contbox10 ulspan:nth-child(1)
    li:nth-child(1) span{
  background-color: rgb(9, 129, 211);
}
.content-boxes #contbox10 ulspan:nth-child(1)
    li:nth-child(2) span{
  background-color: rgb(230, 71, 8);
}
.content-boxes #contbox10 ulspan:nth-child(2)
    li:nth-child(1) span{
  background-color: rgb(199, 5, 134);
}
.content-boxes #contbox10 ulspan:nth-child(2)
    li:nth-child(2) span{
  background-color: rgb(19, 151, 101);
}

.content-boxes p{
```

```
  font-size: 18px;
}

.content-boxes .content-box3 .awardsCont{
    position: relative;
}

.content-boxes .content-box3 .awardsCont span{
  position: absolute;
    visibility: hidden;
    top: 20px;
    background: grey;
    color: #fff;
    width: 100px;
    left: -23px;
    right: 0px;
}
.content-boxes .content-box3 .awardsCont:hover
      span{
  visibility: visible;
}

#contbox4 #heartbeatIcon{
  animation: heartbeat 2s ease infinite;
  width: 39px;
  transform: scale(1);
  color: red;
}
```

```
#contbox6 #faCaretUp{
  position: relative;
  padding-right: 14px;
  color: #039f03;
}

.content-boxes #contbox6 h3{
  font-size: 3em;
}

@keyframes heartbeat {
  0%{
    transform: scale(1);
  }
  25%{
    transform: scale(1.1);
  }
  50%{
    transform: scale(1);
  }
  75%{
    transform: scale(1.2);
  }
  100%{
    transform: scale(1);
  }
}
```

```
.content-boxes .boxInBox span{
  display: inline-block;
  width: 15px;
  height: 15px;
  position: relative;
  top: 4px;
  border: 1px solid transparent;
  border-radius: 3px;
  text-align: center;
}

.content-boxes .boxInBox div{
  cursor: pointer;
}

.content-boxes .boxInBoxdiv:nth-child(1) span{
  background-color:#eb890a ;
}
.content-boxes .boxInBoxdiv:nth-child(2) span{
  background-color:#ba68c8 ;
}
.content-boxes .boxInBoxdiv:nth-child(3) span{
  background-color: chartreuse ;
}
.content-boxes .boxInBoxdiv:nth-child(4) span{
  background-color:#f9105e ;
}
.content-boxes .boxInBoxdiv:nth-child(5) span{
background-color:black ;
```

```
}
.content-boxes .boxInBoxdiv:nth-child(6) span{
  background-color:#0fdfca ;
}
.content-boxes .boxInBoxdiv:nth-child(7) span{
  background-color:#757575 ;
}
.content-boxes .boxInBoxdiv:nth-child(8) span{
  background-color: var(--themeColor) ;
}
.content-boxes .boxInBoxdiv:nth-child(9) span{
  background-color:#FFEB3B ;
}
.content-boxes .boxInBoxdiv:nth-child(10) span{
  background-color:#0b5cff ;
}
.content-boxes .boxInBoxdiv:nth-child(11) span{
  background-color:#ef0784 ;
}

.content-boxes .content-box-common{
  margin: 13px 11px;
  background: #ebebeb;
  border: 1px solid transparent;
  border-radius: 7px;
  box-shadow: 1px 1px 5px 1px #875cd1;
}
```

```
.content-boxes .content-box2{
  text-align: center;
}

.content-box2 .percent{
  width: 50%;
  position: relative;
  transform: rotate(270deg);
  top: 104px;
  right: 102px;
}

.content-boxes .deactive{
  display: none;
}

.content-boxes .active{
  z-index: 100;
  position: fixed;
  display: block;
  width: 77%;
  height: 92vh;
  background-color: rgba(0, 0, 0, .55);
  border-radius: 5px;
}

.content-boxes .active .contentActive{
  background: #fff;
  margin: 30px auto;
```

```
  width: 800px;
  height: 90%;
  border: 1px solid black;
  border-radius: 6px;
  box-shadow: 0px 5px 15px rgba(0, 0, 0, .5);
  animation-name: slidedown;
  animation-duration: .3s;
  animation-fill-mode: both;
  animation-timing-function:  ease-in-out;
}

@keyframes slidedown {
  0%,10%, 90%,100%{
      animation-timing-function:  ease-in-out;
  }
  0%{
      opacity: 0;
      transform: translate3d(0,-70%,0);
  }
  100%{
      transform: translate3d(0,0,0);
  }
}

.active .contentActive #closeActive{
  font-size: 50px;
  float: right;
  padding: 0 13px;
  position: relative;
```

```
  bottom: 13px;
  color: #fff;
  cursor: pointer;
}

.active .contentActive h1{
  font-size: 45px;
  color: #fff;
  background-color: var(--themeColor);
  padding: 20px 0;
  text-align: center;
}

.active .contentActive div p{
    font-weight: bolder;
    font-size: 19px;
    margin: 0 auto;
    width: 500px;
    line-height: 1.5;
}

.active .contentActive div p span{
  display: inline-block;
  width: 145px;
      }
```

8.4 INDEX (js-backend)

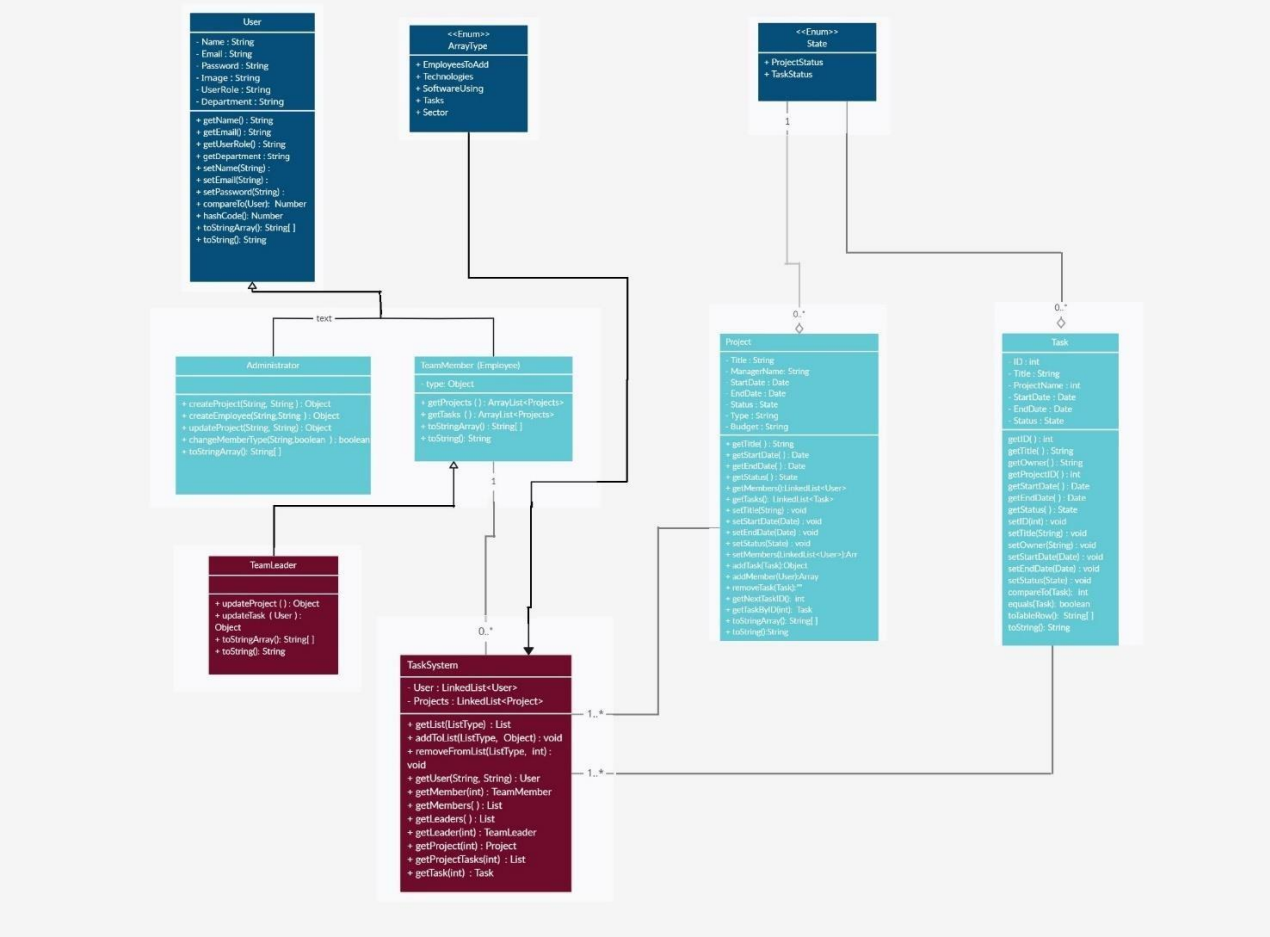
```
      const express = require('express');
      const app = express();
      const mongoose = require('mongoose');
```

```
const adminhandler = require('./admin/adminHandler');
const taskHandler = require('./task/TaskController');
const empHandler = require('./employee/employeeHandler');
const projectHandler = require('./project/projectController');
const otherdataController = require('./otherdata/otherdataContr')
const cors = require('cors');
const empSchema = require('./employee/employeeSchema')

app.use(express.json());
app.use(cors())

// create admin
app.post('/taskmanger/inns/api/admin/createAdmin',  adminhandler.postAdmin);
// login admin
app.post('/taskmanger/inns/api/admin/loginadmin' , adminhandler.loginAdmin);

// add task
app.post('/taskmanger/inns/api/task/addTask',  taskHandler.addTask);
// view task
app.get('/taskmanger/inns/api/task/viewAllTask', taskHandler.viewAllTask);
app.get('/taskmanger/inns/api/task/viewOneTask', taskHandler.viewOneTask);
// update task
app.patch('/taskmanger/inns/api/task/updateTask',  taskHandler.updateTask);
// get count of completed task
app.get('/taskmanger/inns/api/task/countOfCompSts', taskHandler.countOfCompSts);
// get task for emp
app.get('/taskmanager/api/employee/getTaskForEmp',  taskHandler.getTaskForEMp)

// emp
```

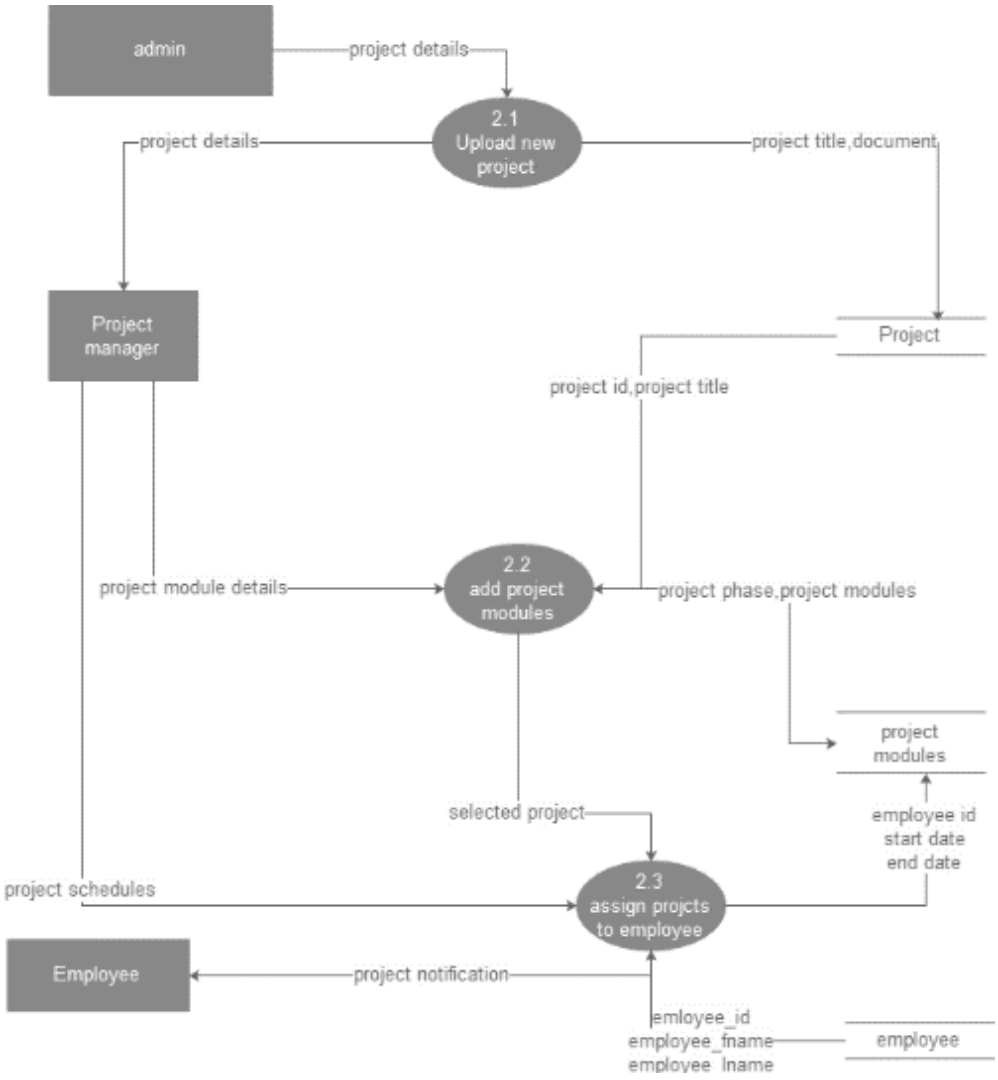
```
app.post('/taskmanager/inns/api/employee/create',empHandler.postEmployee);
app.post('/taskmanager/inns/api/employee/login',empHandler.postempLogin);
// view one emp data
app.get('/taskmanger/inns/api/employee/viewOneEmp',  empHandler.viewOneEmp);
// getEmp username &Emprole
app.get('/taskmanager/api/employee/getUsername',empHandler.getUsername);
// get fn& ln of Emp
// app.get('/taskmanager/api/employee/getNameOfEmp',empHandler.getNameOfEmp);
// get count of Emp
app.get('/taskmanager/api/employee/getCountOfEmp',  empHandler.countOfEmp);
// update emp profile
app.patch('/taskmanager/api/employee/updateEmpProfile',  empHandler.updateEmp);

// add project
app.post('/taskmanger/inns/api/project/addProject',  projectHandler.addProject);
// view project
app.get('/taskmanger/inns/api/project/viewAllProject', projectHandler.viewAllProject);
app.get('/taskmanger/inns/api/project/viewOneProject', projectHandler.viewOneProject);
// get emp name in one project
app.get('/taskmanger/inns/api/project/getEmpInPro',  projectHandler.getEmpInPro);
// get count of project
app.get('/taskmanager/api/project/getCountOfPro',  projectHandler.countOfProject);
// get dead line for project
app.get('/taskmanager/api/project/getDeadline',  projectHandler.getDateOfDeadLine);
// update project
app.patch('/taskmanager/api/project/updateProject',  projectHandler.updateProject);
// getEmp project &Empname in projects
app.get('/taskmanager/api/employee/getProjectname',projectHandler.getProjectname);
// get proj for emp  ====  need to pass query
```

```
app.get('/taskmanager/api/employee/getProjForEmp',  projectHandler.getProjForEMp)


// otherdata (sector, app type, tech, s/w)
app.get('/taskmanger/inns/api/otherdata/getDataInOtherData',otherdataController.viewOtherData)

app.listen(3012, '127.0.0.1' , ()=>{
console.log('server started.');
})

mongoose.connect('mongodb://127.0.0.1:27017/TaskManager',
// { useNewUrlParser: true, useUnifiedTopology: true }
)
.then(() => {
   return empSchema.collection.dropIndex('userRole_1');
})
.then(() => {
console.log('Unique index on userRole field dropped successfully');
console.log('db connected')
})
.catch((error) => {
console.log('Error dropping unique index:', error);
});
```

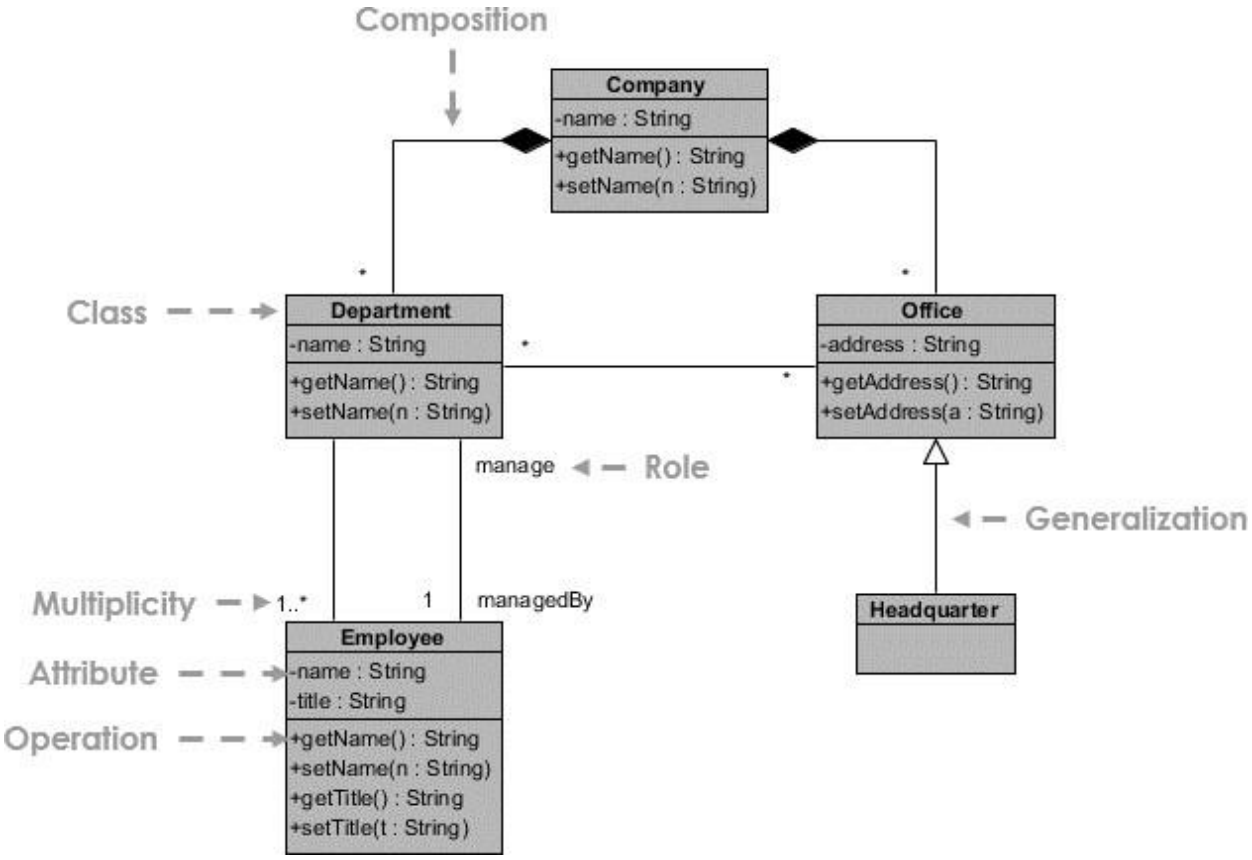
8.5 UML DIAGRAM

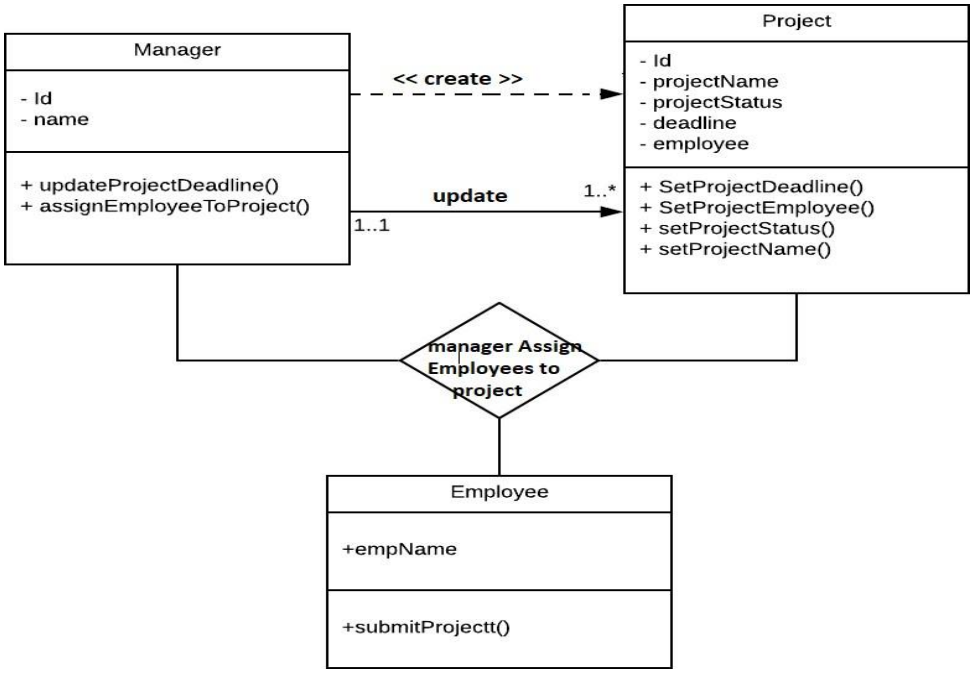


8.6 DATA FLOW DIAGRAM

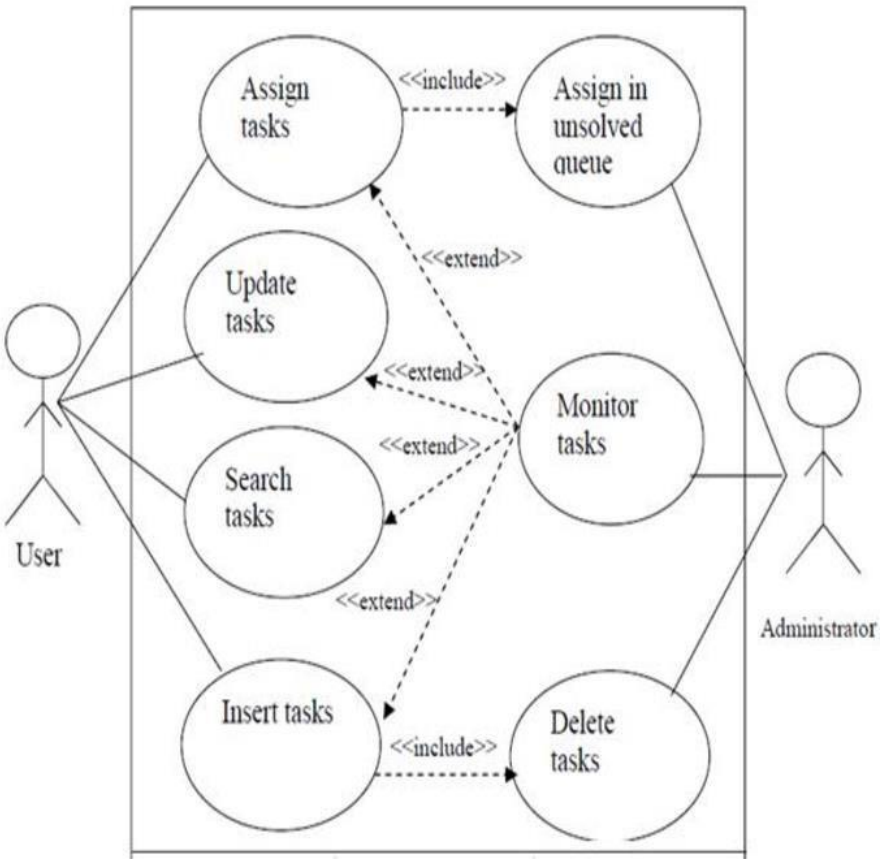


8.7 Activity diagram

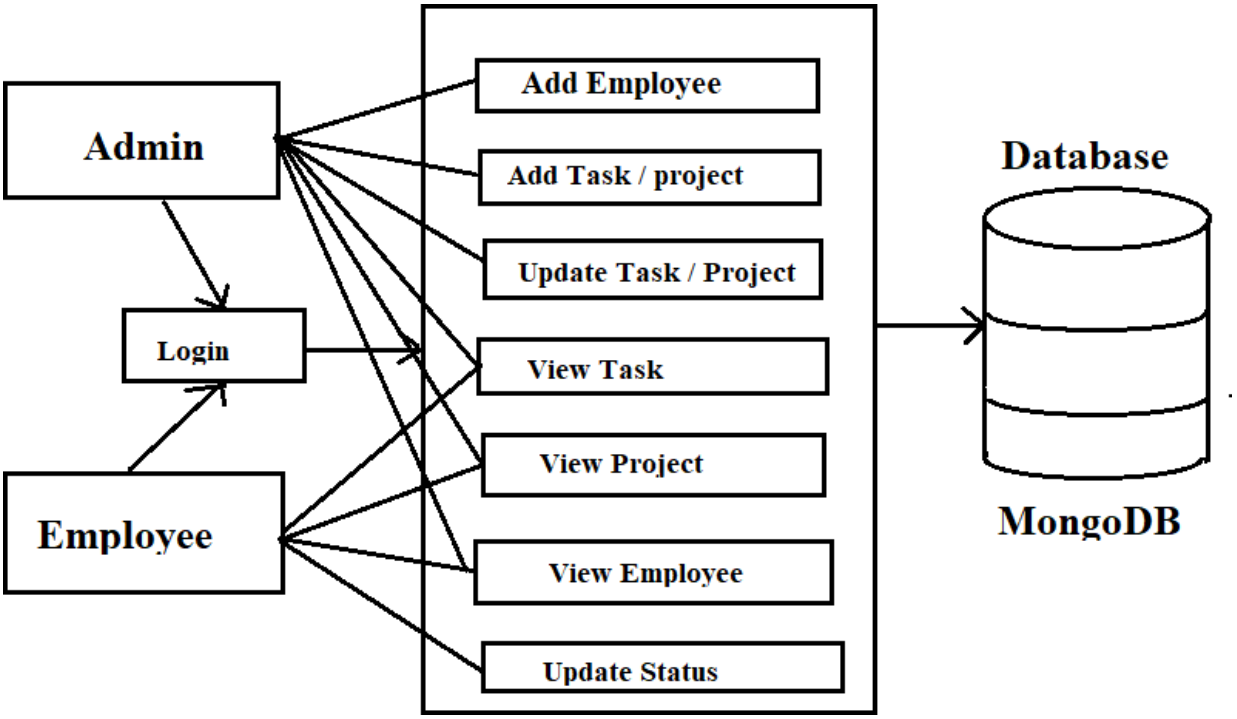




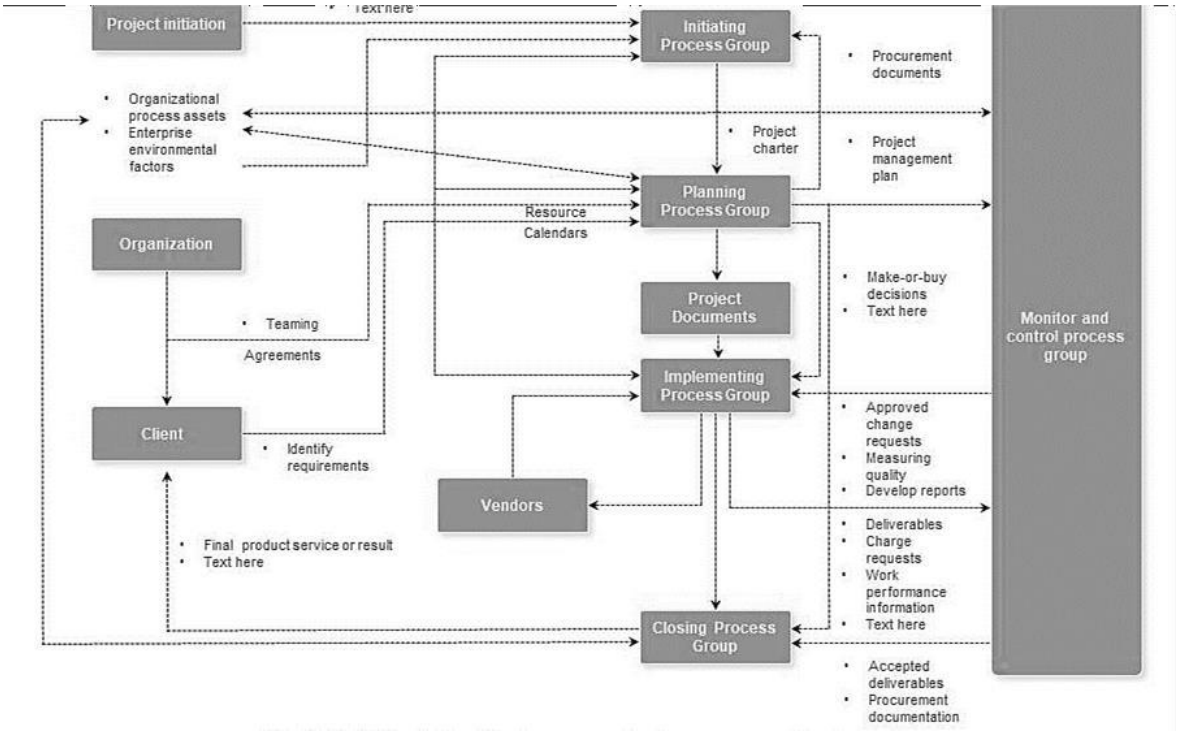
8.8 Use case diagram



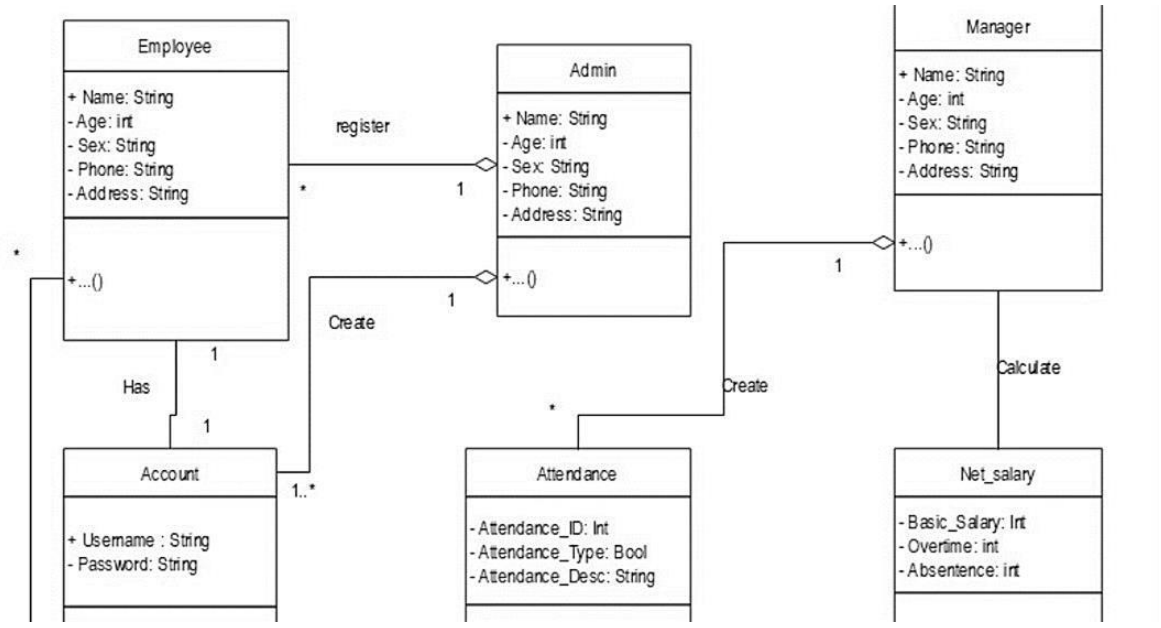
8.9 Component diagram


Admin
Login
Employee
Add Employee
Add Task / project
Update Task / Project
View Task
View Project
View Employee
Update Status
Database
MongoDB

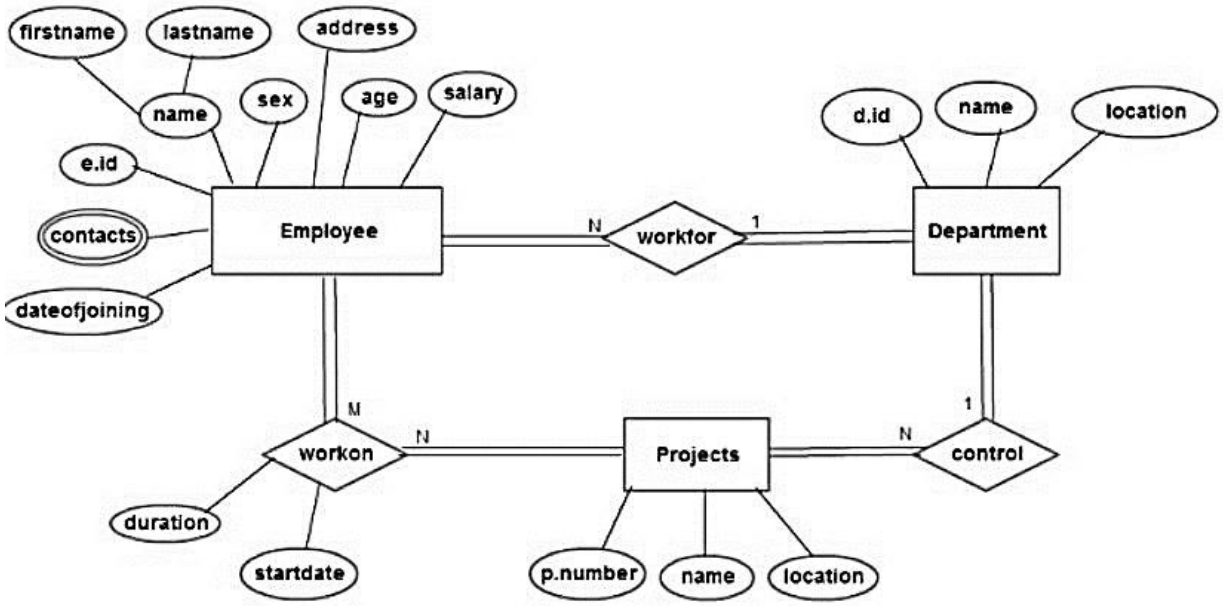
8.10 SEQUENCE DIAGRAM



8.11 DEPLOYMENT DIAGRAM



8.12 ER DIAGRAM



CHAPTER 9

9.1 BIBLIOGRAPHY AND REFERENCES

[1]. Bozikovic, H., Stula, M. (2018). Web design Past, present and future. 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO).

[2]. Carter, B. (2014). HTML Architecture, a Novel Development System (HANDS): An Approach for Web Development. 2014

[3]. Sterling, A. (2019). NodeJS and Angular Tools for JSON-LD. 2019 IEEE 13th 22

[4]. What is Git, Why Should You Use it? (2021)

[5]. Erin Gilliam Haije.(2022). Best Project Management Software and Overview.

[6]. Shahriar Shovon.(2018). Testing REST API Using Postman.

[7]. Yogesh Baiskar, PriyasPaulzagade, KrutikKoradia, Pramod Ingole, DhirajShirbhate.(2022). A Full-Stack Development.

[8]. Sanja Delcev, Drazen Draskovic.(2018).Modern JavaScript frameworks: A Survey Study.

[9]. G. Bierman, M. Abadi, Mads Torgersen.(2014).Understanding TypeScript

[10]. Paxcom.net

[11] .Paymentus.com

[12]. Gitlab.com

10.1 BOOK AND REFERENCES

[1] Android Studio 2 Development Essentials, Book by Neil Smyth.

[2] PHP and MySQL Web Development, Book by Luke Welling, 2001

[3] D. A. Hillson, "Using a Risk Breakdown Structure in project management", Journal Of Facilities Management, vol. 2, no. 1, pp. 85-97, 2013.

[4] S. McKenna, "Organisational Complexity and Perceptions of Task", Task Management: An International Journal, vol. 3, no. 2, pp. 53-64, 2013.

[6] A. Aleshin, "Time and Risk Management of International Projects", International Journal of Project Management, no. 19, pp. 207-222, 2014

[7] J. Ellis, L. Kvavilashvili, "Prioritization of tasks in 2000: Past present and future directions", Task Management, vol. 14, pp. 1-9, 2000.

[8] Drake Baer, "Dwight Eisenhower Nailed AMajor Insight About Productivity", Business Insider, 2014.