

DECLARATION

I affirm that the project work titled “**TASK TRACKING COLLABORATION SYSTEM**” being submitted in partial fulfilment for the award of Master of Computer Applications is the original work carried out by me. It has not formed the part of any other project work submitted for award of any degree or diploma either in this or any other university.

(Signature of the Candidate)

PRIYA A

REGNO:727622MCA007

I certify that the declaration made above by the candidate is true

(Signature of the Guide)

Dr. R. MUTHUSAMI,M.C.A.,M.Phil.,Ph.D

Head & Assistant Professor(SG)/(MCA)

ABSTRACT

Task Tracking Collaboration system technology makes it possible for project managers and team members to jointly develop, allocate, track, and finish tasks. To ensure data security, the application has role-based access control and user authentication. Project development, task assignment, and progress monitoring are key elements. Essential information can be added to projects, such as deadlines, priority, and descriptions. As soon as tasks are assigned, team members are notified and have the ability to update the status in real time. In order to help in decision-making and resource allocation, the system offers a visual depiction of project progress. Its incorporation of user-friendly interfaces, automated notifications, and project visualization helps projects succeed and increase productivity.

**Dr. MAHALINGAM COLLEGE OF ENGINEERING AND
TECHNOLOGY**

POLLACHI – 642003

DEPARTMENT OF COMPUTER APPLICATIONS (MCA)

MAJOR PROJECT REPORT

APRIL 2024

This is Certify that the project entitled

TASK TRACKING COLLABORATION SYSTEM

is the Bonafide record of project work done by

PRIYA A

REG NO: 727622MCA007

of Master of Computer Applications during the year 2022-2024

Project guide

Dr. R. MUTHUSAMI,MCA.,M.Phil.,Ph.D

Head & Assistant Professor(SG)/MCA

Head the Department

Dr. R. MUTHUSAMI,MCA.,M.Phil.,Ph.D

Head & Assistant Professor(SG)/MCA

Submitted for the Project Viva-Voice examination held on_____

Internal Examiner

External Examiner

ACKNOWLEDGEMENT

I express my gratitude to **Dr. C.RAMASWAMY, M.E., F.I.V., Ph.D., Secretary, NIA Educational Institutions, Pollachi**, for having provided me the facilities to do the project successfully.

I express my sincere thanks to **Dr. P.GOVINDASAMY, B.E., M.E, Tech., Ph.D., Principal**, Dr. Mahalingam College Of Engineering And Technology, Pollachi, for having provided me the facilities to do the project successfully.

It is a great privilege and pleasure for me to express my gratitude to **Dr. A. SENTHIL KUMAR, B.E., M.E., Ph.D., Dean (Academic and Autonomous), Dr. S. RAMAKRISHNAN, B.E., M.E., Ph.D., Dean (Research & Innovation), Dr. CALVIN SOPHISTUS KING, M.Tech., Ph.D, Dean (Industry Relations and Talent Development)**, Dr. Mahalingam College of Engineering and Technology, Pollachi, for persistent encouragement..

I own deep sense of gratitude to **Dr. R. MUTHUSAMI, MCA., M.Phil., Ph.D., Head of Department of Computer Applications** for appreciating my goal. I express my sincere thanks for him for his constant encouragement.

I express my thanks to my guide **Dr. R. MUTHUSAMI, MCA., M.Phil., Ph.D., Head & Assistant Professor Department of Computer Applications** for her valuable guidance and support to meet the successful completion of my project.

I express my sincere thanks to all staff members of Department of Computer Applications for their encouragement and valuable guidance throughout this project.

Last but not the least, I would like to thank **my family** and **my friends** for putting up with me spending so much time providing encouragement and valuable suggestions, throughout the project tenure.

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	Iv
	LIST OF FIGURES	Ix
	LIST OF ABBREVIATIONS	Iv
1	INTRODUCTION	1
	1.1 Objectives of Project	1
2	SYSTEM ANALYSIS	2
	2.1 Existing System	2
	2.2 Proposed System	2
	2.2.1 Advantages of Proposed System	3
	2.3 Requirement Gathering	3
	2.3.1 Functional Requirement	3
	2.3.2 Non-Functional Requirement	4
	2.4 Feasibility Study	4
	2.4.1 Economical Feasibility	4
	2.4.2 Operational Feasibility	4
	2.4.3 Technical Feasibility	4
3	SYSTEM SPECIFICATION	5
	3.1 Hardware Specification	5
	3.2 Software Specification	5
4	SOFTWARE DESCRIPTION	6
	4.1 Programming Languages	6
	4.2 Development Tools and Technologies	6
	4.2.1 Front End	6
	4.2.2 Back End	9

5	PROJECT DESCRIPTION	10
	5.1 Problem Definition	10
	5.2 Objective of Project	10
	5.3 Module Description	10
	5.4 System Design	11
	5.4.1 System Flow Diagram	11
	5.4.2 Data Flow Diagram	12
	5.4.3 Use Case Diagram	14
	5.5 Input Design	15
	5.6 Output Design	15
6	SYSTEM TESTING	16
	6.1 Testing Strategies	17
	6.1.1 Unit Testing	17
	6.1.2 Integration Testing	17
	6.1.3 Validation Testing	17
	6.2 Test Cases	18
7	SYSTEM IMPLEMENTATION	19
	7.2 System Implementation	19
	7.3 System Maintenance	19
8	CONCLUSION AND FUTURE ENHANCEMENTS	21
	8.1 Conclusion	21
	9.2 Scope for Future Enhancement	21
9	APPENDICES	22
	9.1 Source code	22
	9.2 Screen shots	32
10	REFERENCES	



TASK TRACKING COLLABORATION SYSTEM



MINI PROJECT - II

Submitted by

NAME: PRIYA A

REG NO: 727622MCA007

*In partial fulfillment of the requirement for the
Award of the degree of*

MASTER OF COMPUTER APPLICATIONS

**Dr. MAHALINGAM COLLEGE OF ENGINEERING AND
TECHNOLOGY POLLACHI – 642003**

**(Approved by AICTE, affiliated to Anna University and Accredited by
NAAC with “A++” Grade)**

APRIL 2024

CHAPTER 1

INTRODUCTION

1.1 OBJECTIVIES

A task tracking system serves as a cornerstone for efficient project management, offering a comprehensive solution to the challenges of task organization, monitoring, and coordination. By providing a centralized repository for all tasks and projects, it ensures clarity, accountability, and alignment across teams and individuals. The system allows users to categorize tasks based on project, priority, deadlines, or other relevant criteria, facilitating effective resource allocation and workload management.

Moreover, task tracking systems empower teams to collaborate seamlessly by facilitating communication, feedback, and knowledge sharing. Team members can assign tasks, share updates, and collaborate on documents or files within the platform, fostering a culture of transparency and teamwork. This collaborative environment not only enhances productivity but also promotes innovation and creativity as team members can leverage each other's expertise and insights.

One of the key benefits of task tracking systems is their ability to provide real-time visibility into project progress and performance. Through customizable dashboards, reports, and analytics, stakeholders can gain insights into task completion rates, bottlenecks, resource utilization, and overall project health. This data-driven approach enables informed decision-making, timely interventions, and proactive risk management, ultimately driving project success.

CHAPTER 2

SYSTEM ANALYSIS

System analysis is a general term that can refer to stored process for identifying and solving problem. Analysis implies the process of breaking something down into part so that the whole may be understood. The definition of system analysis not only the process of analysis but also that of synthesis, which is the process of putting parts together to form a new whole.

2.1 EXISTING SYSTEM

The existing system is a manual system. As there process is done manually it takes long time to prepare all the report regarding the daily and monthly reports.

2.1.1 Drawbacks

The existing system manual system is not efficient due to the following reasons.

- Departments manage their own databases separately.
- Define clear workflows outlining how tasks are created, assigned, executed, and monitored within the collaborative environment.
- Teams can't communicate within the platform, share files, and work together on tasks and projects.
- Encourage transparency by regularly updating task statuses, sharing progress reports, and addressing any challenges openly.
- Reporting and analytics are limited and often require manual data extraction and analysis.
- Members have limited access to information from other departments.

2.2 PROPOSED SYTEM

The system is especially developed for making the process fast and without any error occurrence. The above information can be maintained and retrieved from the proposed system. It has been decided to the computerization of the process in the proposed system except with data entry and devices getting no other human interventions are necessary.

2.2.1 FEATURES

Robust Dashboard:

- Robust dashboard with all important charts with real-time reports and insights.

Collaboration and Communication:

- Built-in communication tools for team collaboration, such as comments, mentions, and threaded discussions on tasks.

Prioritization and Sorting:

- Sorting options to arrange tasks by due date, priority, status, or other parameters for better organization.

Progress Tracking and Reporting:

- Generate reports and analytics to analyze task performance, completion rates, and team productivity.

2.3.1 FUNCTIONAL REQUIREMENTS

- The system must support management of access, views, functionality and security roles such as users, and administrator.
- The system provide the ability to preview content prior to being published.
- The system supports custom fields.
- The system must allow downloadable content for users in various types from computer or mobile device.
- The system must alert administrators of a reasonable accommodation requirement

2.3.2 NON-FUNCTIONAL REQUIREMENTS

- **Security:** The system must be secure from unauthorized access.
- **Performance:** The system must be able to handle the required number of users without any degradation in performance.
- **Portability:** The system must be able to run on different platforms with minimal changes.
- **Reliability:** The system must be reliable and meet the requirements of the user.

2.4 FEASIBILITY STUDY

The feasibility of the project is analysed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential. Three key considerations involved in the feasibility analysis are:

2.4.1 ECONOMIC FEASIBILITY

This study is carried out to check the economic impact will have on the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus, the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products have to be purchased.

2.4.2 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes for the implementing this system.

2.4.3 OPERATIONAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it.

CHAPTER 3

SYSTEM REQUIREMENTS

3.1 HARDWARE SPECIFICATION

Processor : Intel i5 & above(3.5 GHz speed)
Hard disk : 1 TB
RAM : 8GB

3.2 SOFTWARE SPECIFICATION

Operating System : Windows 10 pro(64 bit)
IDE : Visual Studio 2019
Front End : Angular 17.1.0
Back End : Nodejs 20.11.1
Database : MySQL 8.0

CHAPTER 4

SOFTWARE DESCRIPTION

4.1 PROGRAMMING LANGUAGES

4.1.1 FRONT END – ANGULAR

Angular is a platform and framework for building single-page client applications using HTML and Type Script. Angular is written in Type Script. It implements core and optional functionality as a set of Type Script libraries that you import into your applications.

The architecture of an Angular application relies on certain fundamental concepts. The basic building blocks of the Angular framework are Angular components that are organized into Ng Modules. Ng Modules collect related code into functional sets; an Angular application is defined by a set of Ng Modules. An application always has at least a root module that enables bootstrapping, and typically has many more feature modules.

- Components define views, which are sets of screen elements that Angular can choose among and modify according to your program logic and data
- Components use services, which provide specific functionality not directly related to views. Service providers can be injected into components as dependencies, making your code modular, reusable, and efficient.

Modules, components and services are classes that use decorators. These decorators mark their type and provide metadata that tells Angular how to use them.

- The metadata for a component class associates it with a template that defines a view. A template combines ordinary HTML with Angular directives and binding Mark up that allow Angular to modify the HTML before rendering it for display.
- The metadata for a service class provides the information Angular needs to make it available to components through dependency injection (DI)

An application's components typically define many views, arranged hierarchically. Angular provides the Router service to help you define navigation paths among views. The router provides sophisticated in-browser navigational capabilities.

Modules

Angular Ng Modules differ from and complement JavaScript (ES2015) modules. An Ng Module declares a compilation context for a set of components that is dedicated to an application domain, a workflow, or a closely related set of capabilities. An Ng Module can associate its components with related code, such as services, to form functional units.

Every Angular application has a root module, conventionally named App Module, which provides the bootstrap mechanism that launches the application. An application typically contains many functional modules.

Like JavaScript modules, Ng Modules can import functionality from other Ng Modules, and allow their own functionality to be exported and used by other Ng Modules. For example, to use the router service in your app, you import the Router Ng Module.

Organizing your code into distinct functional modules helps in managing development of complex applications, and in designing for reusability. In addition, this technique lets you take advantage of lazy-loading that is, loading modules on demand to minimize the amount of code that needs to be loaded at start up.

Templates, directives, and data binding

A template combines HTML with Angular Markup that can modify HTML elements before they are displayed. Template directives provide program logic, and binding Markup connects your application data and the DOM.

Components

Every Angular application has at least one component, the root component that connects a component hierarchy with the page document object model (DOM). Each component defines a class that contains

application data and logic, and is associated with an HTML template that defines a view to be displayed in a target environment.

The `@Component()` decorator identifies the class immediately below it as a component, and provides the template and related component-specific metadata.

Services and dependency injection

For data or logic that isn't associated with a specific view, and that you want to share across components, you create a *service* class. A service class definition is immediately preceded by the `@Injectable()` decorator. The decorator provides the metadata that allows other providers to be injected as dependencies into your class.

Dependency injection (DI) lets you keep your component classes lean and efficient. They don't fetch data from the server, validate user input, or log directly to the console; they delegate such tasks to services.

Routing

The Angular Router Ng Module provides a service that lets you define a navigation path among the different application states and view hierarchies in your application. It is modeled on the familiar browser navigation conventions:

- Enter a URL in the address bar and the browser navigates to a corresponding page
- Click links on the page and the browser navigates to a new page
- Click the browser's back and forward buttons and the browser navigates backward and forward through the history of pages you've seen

The router maps URL-like paths to views instead of pages. When a user performs an action, such as clicking a link, that would load a new page in the browser, the router intercepts the browser's behavior, and shows or hides view hierarchies.

If the router determines that the current application state requires particular functionality, and the module that defines it hasn't been loaded, the router *can* lazy-load the module on demand.

The router interprets a link URL according to your application's view navigation rules and data state. You can navigate to new views when the user clicks a button or selects from a drop box, or in response

To define navigation rules, you associate *navigation paths* with your components. A path uses a URL like syntax that integrates your program data, in much the same way that template syntax integrates your views with your program data. You can then apply program logic to choose which views to show or to hide, in response to user input and your own access rules.

4.1.2 BACK END – NodeJs

Node.js, the backend refers to the server-side portion of the application that handles data processing, business logic, and interacts with databases or external services. Node.js is a popular runtime environment for building scalable and efficient backend systems using JavaScript. Here are some key aspects of developing the backend in a Node.js project.

Server Configuration: Setting up a Node.js server using frameworks like Express.js, Koa.js, or Nest.js. These frameworks provide features for routing, middleware, error handling, and other server-related tasks, making it easier to build robust backend applications. API Development: Defining and implementing RESTful or Graph APIs to expose functionality and data to clients (such as web browsers, mobile apps, or other services). This involves creating routes, handling HTTP requests and responses, and serializing/deserializing data in JSON format. Database Integration: Connecting to databases like MongoDB, MySQL, PostgreSQL, or others to store and retrieve data. Node.js provides various database drivers and ORMs (Object-Relational Mapping) libraries to interact with databases efficiently. ORM libraries like Mongoose for MongoDB or Sequelize for SQL databases simplify database operations by providing a higher-level abstraction.

Authentication and Authorization: Implementing user authentication and authorization mechanisms to secure endpoints and control access to resources. This may involve using techniques like JWT (JSON Web Tokens), OAuth, or session-based authentication, along with middleware for validating user credentials and managing sessions.

CHAPTER 5

PROJECT DESCRIPTION

5.1 MODULES DESCRIPTION:

This automation consists of two kind of users namely admin and user. Both can communicate with each other to share the required document for according to their needs. User can be updated and also modified their details in future.

Admin Module:

- Manage/update user details
- View user details
- Upload Task details

User Module:

- Login
- Register
- View Task details
- Upload task materials

Admin Dashboard:

In these sections admin have briefly view the total number of users and their details.

User Side:

User have to login the page it will move to the next page. User have to view task details, title, description, start date, due date.

5.2 USECASE DIAGRAM

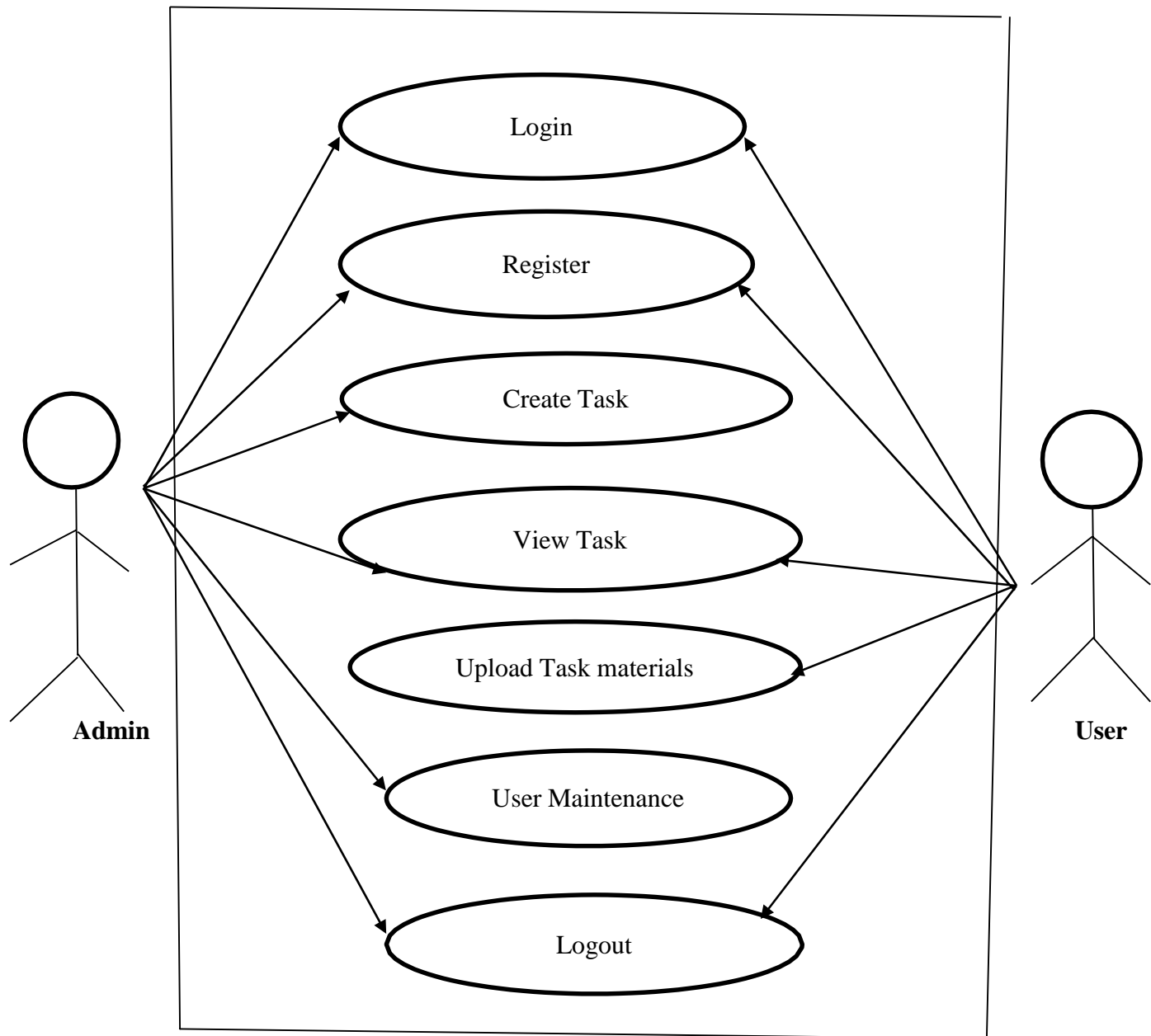


Figure 5.2.1-Usecase diagram

5.1.1 DATABASE DESIGN

The data pertaining to proposed system is voluminous that a careful design of the database must Proceed design of the database must proceed before storing the data in the database. A database management system provides flexibility in the storage and retrieval of data held production of information. The DBMS is a bridge between the application programs. This determines what data are needed and how they are processed and the operating system of the computer, which is responsible for placing data on the magnetic storage devices. A schema defines the database and a sub scheme defines the portion of the database that a specific program will use.

Table Name : Admin_Details

Description : This table is to store the admin entry Details.

Primary key : a_id

Field	Type	Size	Constraints	Description
a_id	varchar	11	Primary key	Describes admin id
Admin name	varchar	10	Not null	Specifies the admin name
User name	varchar	20	Not null	Specifies the admin user name
password	varchar	10	Not null	Password
Email	varchar	25	Not null	Describe the email of admin

Table No.: 5.5.1 Admin_Details

Table Name : Security_Details(Employees)

Description : This table is to store the security list Details.

Primary key : e_id

Field	Type	Size	Constraints	Description
e_id	varchar	20	Primary key	Describes security ID
name	varchar	30	Not null	Specifies the security name
password	varchar	10	Not null	Denotes the password
username	varchar	20	Not null	Security username

Table No.: 5.5.2 Security_Details

LEVEL 0

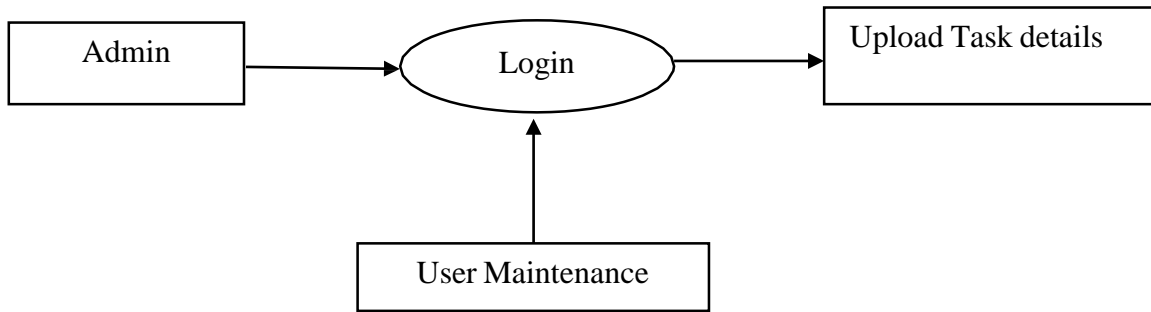


Figure 5.2.2-Level 0 DFD

LEVEL 1

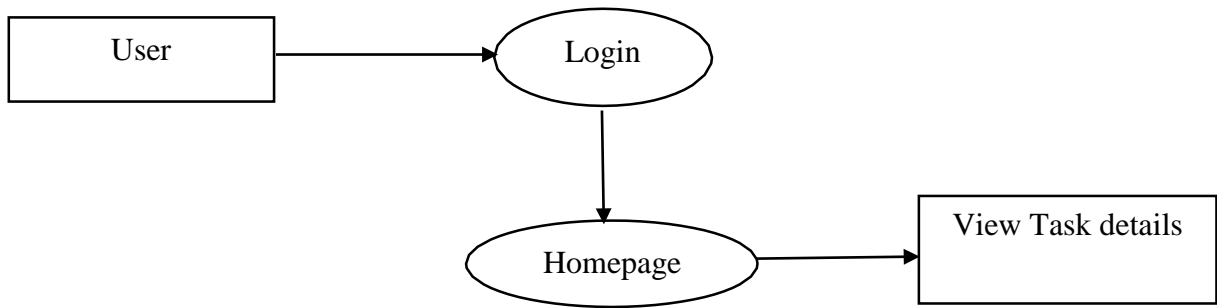


Figure 5.2.3-Level 1 DFD

LEVEL 2

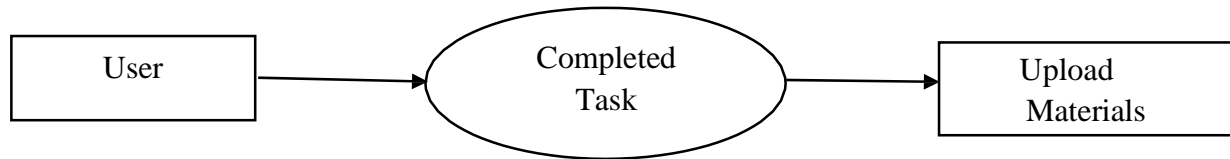
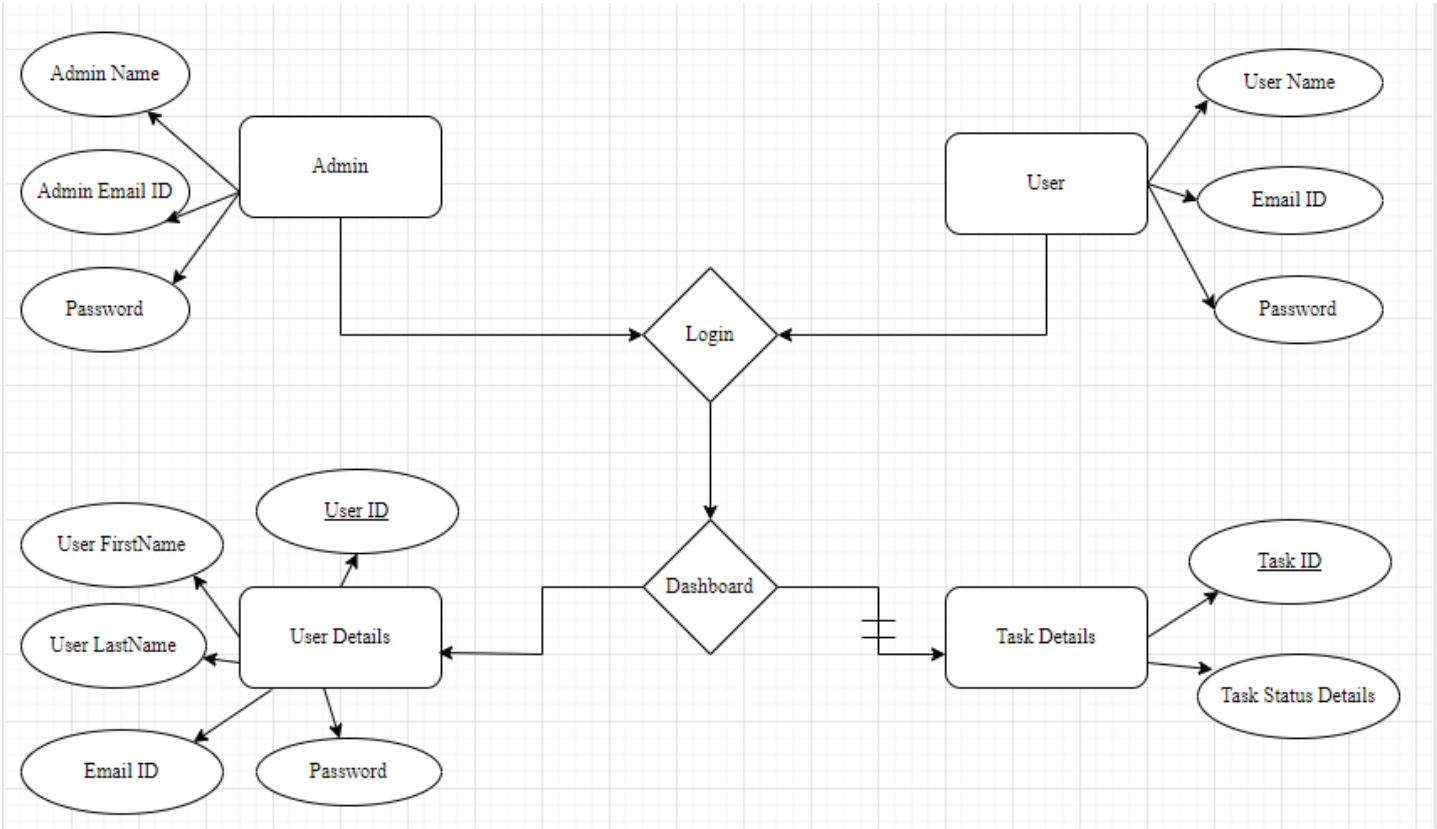


Figure 5.2.4-Level 1DFD

5.1.2 ENTITY RELATIONSHIP DIAGRAM



5.1.2 ENTITY RELATIONSHIP DIAGRAM

5.2.3 SYSTEM FLOW DIAGRAM

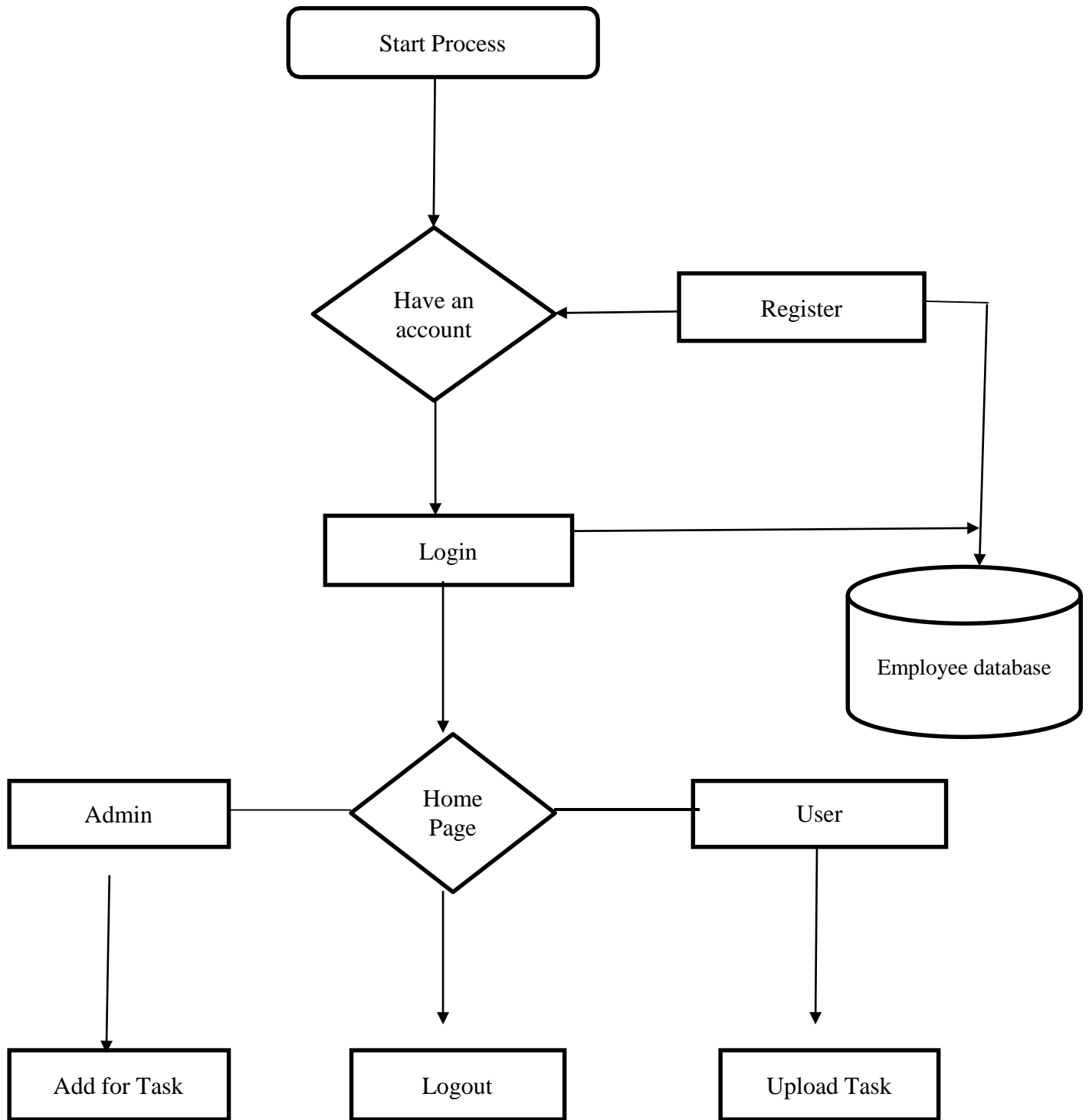


Figure 5.2.3 SYSTEM FLOW DIAGRAM

5.6 INPUT DESIGN

Input design is one of the most expensive phases of the operation of computerized system and is often the major problem of a system. A larger number of problems with a system can usually be traced back to fault input design and method. Needless to say, therefore, that the input data is the life block of a system and has to be analyzed and designed with the most consideration.

Login Details

The login menu allows the user to enter according to the valid username and password. Login detail is created using controls as label boxes and text boxes to the valid username and password and a button control is used to validate the login.

5.7 OUTPUT DESIGN

Output design generally refers to the results reports that are generated by the system. For many end-users, output is the main reason for developing the system and the basis on which they evaluate the usefulness of application. The objective of a system finds its shape in terms of the output. The analysis of the objective of a system leads to determination of outputs. Outputs of a system can take various forms. The users of the output, its purpose and sequence of details to be printed are the factors to be considered. The basic requirement of output is that it should be accurate, timely and appropriate, in terms of content, medium and layout for its intended purpose.

CHAPTER 6

SYSTEM TESTING

Testing is the process of confirming that a program or system does what it is proposed off, Testing is the only way to assure the quality of software and it is an umbrella activity rather than a separate phase. This is an activity to be performed in parallel with the software efforts and one that consists of its own phase of analysis, design, implementation, execution and maintenance.

During analysis and design, a software verification plan and an acceptance test plan is prepared. The verification plan describes the methods to be used in verifying that the requirements are satisfied by the design documents and that the source is consistent with the requirements specification and design documents. The acceptance test plan includes test cases, outcomes and capabilities demonstrated by each test case. Following completion of the verification plan and acceptance plan, a software verification review is held to evaluate the adequacy of the plans.

Prior to product delivery, a functional audit and a physical audit are performed. The functional audit reconfirms that the requirements have been met, the physical audit verifies that the source code and all associated documents are complete, consistent with one another and ready to deliver. A software verification summary is prepared to describe the results of all reviews.

6.1 TESTING METHODOLOGIES

6.1.1 Unit Testing

This testing method considers a module as single unit and checks the unit at interfaces and communities with other modules rather than getting into details at statement level. Here the module will be treated as black box, which will take some inputs and generate output. Outputs for a given set of input combination are pre calculated and are generated by the module.

6.1.2 Integration Testing

Here all the pre-tested individual modules will be assembled to create a larger system and tests are carried out at system level to make sure that all modules are working with each other. This testing methodology helps in making sure that all modules which are running perfectly when checked individually and cases to check all modules once and then a generated test combination of test paths with the system to make sure that no path is making its way into chaos.

6.1.3 Validation Testing

Testing is major quality control measure employed during software development. Its basic function is to detect errors. Sub functions when combined may not produce than it is desired. Global data structures can represent the problems. To uncover errors that are associated with interfacing the objective is to make test modules and built a program structure that has detected by design. In a non-incremental integration all the modules are combined in advance and the program is tested as whole. Here error will appear in an end-less loop function.

TEST CASE

TEST CASE ID	TEST CASE NAME	TEST CASE DESCRIPTION	TEST PRIPORITY		RESULT
			STEP	RESULT	
01	Username	To Verify the username if it is unique.	Nothing enters and focuses on next control.	An error message shows user name is required	Pass
02	Password	To verify the password should be unique	Nothing enters and focuses on next control	Accepted	Pass
03	Add admin	To ensure that a admin detail is added to the system successfully	If admin already exists in the system,	Error Message should display.	Pass
04	Edit Details of admin	To ensure that once different details are provided on the edit personal details form and submitted, these details are altered in the database to reflect the recent changes.	When the form is altered the details should be altered in the database	Accepted	Pass
05	Add employee	To ensure that employee is added to system successfully	Nothing enters and focuses on next control.	Error Message should display	Pass
06	Edit Details of Employee	To ensure that once different details are provided on the edit personal details form and submitted, these details are altered in the database to reflect the recent changes.	When the form is altered the details should be altered in the database	Accepted	Pass

CHAPTER 7

SYSTEM IMPLEMENTATION

7.1 OVERVIEW OF TECHNOLOGY USE

This application can be entered using a username and password as we discussed earlier. We also describe about the potential uses of this system. The interface of the software is user friendly.

7.2 SOFTWARE

In our application, we use Angular 15, API to execute a simple and effective reminder application. We make our interface easily understandable and also make it simple as much as possible for future maintenance by the User.

7.3 SYSTEM IMPLEMENTATION

Bootstrap is a free and open-source CSS framework directed at responsive, mobile-first front-end web development. It contains HTML, CSS and JavaScript-based design templates for typography, forms, buttons, navigation, and other interface components.

Step1: As a result, to get started with Angular, you will need to have Node.js installed on your system. You can head to the NodeJS official website to download the software. Install the latest version and confirm them on you command prompt by running the following commands:

```
node--version
```

```
npm--v
```

Step2: To install the CLI, in the command prompt, type the following commands

Installation:

```
npm install -g @angular/cli
```

Confirmation – `ng--version`

Step3: Create a folder for your application in the desired location on your system and open it on VSCode. Open a new terminal and type in the following command to create your app folder.

```
ng create projectname
```

Step 4:To run the application, change the directory to the folder created, and use the `ng` command.

```
cd hello-world
```

```
ng serve
```

Once run, open your browser and navigate to `localhost:4200`. If another application is running on that address, you can simply run the command.

```
ng serve--port
```

7.4 SYSTEM MAINTENANCE

The implementation view of software requirements presents the real-world manifestation of processing function and information structures. In some case, a physical representation is developed as the first step in software design. The analyst must recognize the constraints imposed by predefined system elements in and consider the implementation view of function and information when such view is inappropriate.

Implementation Plan

System implementation is the process of making the newly designed system fully operational. The system is implemented after careful testing. Implementation is a stage in the project where theoretical design is turned into working system in order to maximize efficiency and productivity. The most critical stage in achieving a new system is in getting the approval from the system manager. The newly designed system put into work process, after the testing is over. The system will be implemented in phase along with existing system and it take over happens when the full testing is defined to be perfect, till then the parallel run is done.

Implementation procedures are

- Implementation simply means converting a new computer system to replace an existing one.
- Implementation of computer system to replace a manual system.

After the data has been initially set, the system is ready for use. The change over from the existing system to new system is a step-by-step process.

CHAPTER 8

CONCLUSION AND FUTURE ENHANCEMENTS

CONCLUSION

Task Tracking Collaboration System applications are invaluable tools for individuals and teams looking to improve productivity, organization, and collaboration. By centralizing tasks, streamlining processes, and providing visibility into project progress, these applications empower users to work more efficiently and effectively.

FUTURE ENHANCEMENTS

The future enhancements are possible to implement along with this Task tracking app has been designed at the maximum possible excellence. In future the application is enhanced with the possibilities of the all poultry farm information are known about the app. This application can be extended easily without affecting the functionality.

- Users have to discuss their team members using Chabot.
- Admin have to conducting team meetings.
- Task status are viewed in graphical representation.

APPENDICES

9.1 SOURCE CODE

login.component.html

```
<div class="card">
  <h4 class="card-header">Login</h4>
  <div class="card-body">
    <form [formGroup]="form" (ngSubmit)="onSubmit()">
      <div class="mb-3">
        <label class="form-label">Username</label>
        <input type="text" formControlName="username" class="form-control" [ngClass]="{
'is-invalid': submitted && f.username.errors }" />
        <div *ngIf="submitted && f.username.errors" class="invalid-feedback">
          <div *ngIf="f.username.errors.required">Username is required</div>
        </div>
      </div>
      <div class="mb-3">
        <label class="form-label">Password</label>
        <input type="password" formControlName="password" class="form-control"
[ngClass]="{ 'is-invalid': submitted && f.password.errors }" />
        <div *ngIf="submitted && f.password.errors" class="invalid-feedback">
          <div *ngIf="f.password.errors.required">Password is required</div>
        </div>
      </div>
      <div>
        <button [disabled]="loading" class="btn btn-primary">
          <span *ngIf="loading" class="spinner-border spinner-border-sm me-1"></span>
          Login
        </button>
        <a routerLink="../register" class="btn btn-link">Register</a>
      </div>
    </form>
  </div>
</div>
```

login.component.ts

```
import { Component, OnInit } from '@angular/core';
import { Router, ActivatedRoute } from '@angular/router';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
import { first } from 'rxjs/operators';
import { AccountService, AlertService } from '@app/_services';
@Component({ templateUrl: 'login.component.html' })
export class LoginComponent implements OnInit {
  form!: FormGroup;
  loading = false;
  submitted = false;

  constructor(
    private formBuilder: FormBuilder,
    private route: ActivatedRoute,
    private router: Router,
    private accountService: AccountService,
    private alertService: AlertService
  ) { }

  ngOnInit() {
    this.form = this.formBuilder.group({
      username: ['', Validators.required],
      password: ['', Validators.required]
    });

    // convenience getter for easy access to form fields
    get f() { return this.form.controls; }

    onSubmit() {
      this.submitted = true;

      // reset alerts on submit
      this.alertService.clear();
    }
  }
}
```

```

    // stop here if form is invalid
    if (this.form.invalid) {
        return;
    }

    this.loading = true;
    this.accountService.login(this.f.username.value, this.f.password.value)
    .pipe(first())
    .subscribe({
        next: () => {
            // get return url from query parameters or default to home page
            const returnUrl = this.route.snapshot.queryParams['returnUrl'] || '/';
            this.router.navigateByUrl(returnUrl);
        },
        error: error => {
            this.alertService.error(error);
            this.loading = false;
        }
    });
}
}

```

registercomponent.html

```

<div class="card">
    <h4 class="card-header">Register</h4>
    <div class="card-body">
        <form [formGroup]="form" (ngSubmit)="onSubmit()">
            <div class="mb-3">
                <label class="form-label">First Name</label>
                const returnUrl = this.route.snapshot.queryParams['returnUrl'] || '/';
                this.router.navigateByUrl(returnUrl);
            </div>
        </form>
    </div>
</div>

```

```

    <div *ngIf="submitted && f.password.errors" class="invalid-feedback">
        <div *ngIf="f.password.errors.required">Password is required</div> div
    *ngIf="f.password.errors.required">Password is requir>
        <div *ngIf="f.password.errors.minlength">Password must be at least 6
characters</div>
    </div>

    <div *ngIf="submitted && f.firstName.errors" class="invalid-feedback">
        <div *ngIf="f.firstName.errors.required">First Name is required</div>
    </div>
</div>
<div class="mb-3">
    <label class="form-label">Last Name</label>
    <input type="text" formControlName="lastName" class="form-control"
[ngClass]="{ 'is-invalid': submitted && f.lastName.errors }" />
    <div *ngIf="submitted && f.lastName.errors" class="invalid-feedback">
        <div *ngIf="f.lastName.errors.required">Last Name is required</div>
    </div>
</div>
<div class="mb-3">
    <label class="form-label">Username</label>
    <input type="text" formControlName="username" class="form-control"
[ngClass]="{ 'is-invalid': submitted && f.username.errors }" />
    <div *ngIf="submitted && f.username.errors" class="invalid-feedback">
        <div *ngIf="f.username.errors.required">Username is required</div>
    </div>
</div>
<div *ngIf="submitted && f.password.errors" class="invalid-feedback">
    <div *ngIf="f.password.errors.required">Password is required</div>
    <div *ngIf="f.password.errors.minlength">Password must be at least 6
characters</div>
</div>
<div>

```

```

        <button [disabled]="loading" class="btn btn-primary">
            <span *ngIf="loading" class="spinner-border spinner-border-sm me-
1"></span>
            Register
        </button>
        <a routerLink=" ../login" class="btn btn-link">Cancel</a>
    </div>
</form>
</div>
</div>

```

Register.component.ts

```

import { Component, OnInit } from '@angular/core';
import { Router, ActivatedRoute } from '@angular/router';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
import { first } from 'rxjs/operators';
import { AccountService, AlertService } from '@app/_services';
@Component({ templateUrl: 'register.component.html' })
export class RegisterComponent implements OnInit {
    form!: FormGroup;
    loading = false;
    submitted = false;

    private formBuilder: FormBuilder,
        private route: ActivatedRoute,
        private router: Router,
        private accountService: AccountService,
    ) { }

    ngOnInit() {
        this.form = this.formBuilder.group({
            firstName: ['', Validators.required],
            lastName: ['', Validators.required],

```

```

        username: ['', Validators.required],
        password: ['', [Validators.required, Validators.minLength(6)]]
    });
}

// convenience getter for easy access to form fields
get f() { return this.form.controls; }

onSubmit() {
    this.submitted = true;

    // reset alerts on submit
    this.alertService.clear();

    // stop here if form is invalid
    if (this.form.invalid) {
        return;
    }

    this.loading = true;

    this.accountService.register(this.form.value)
        .pipe(first())
        .subscribe({
            next: () => {
                this.alertService.success('Registration successful', {
keepAfterRouteChange: true });
                this.router.navigate(['../login'], { relativeTo: this.route });
            },

```

9.2 SCREEN SHOTS

Login Page

Description:

Admin can login using mail-id and password

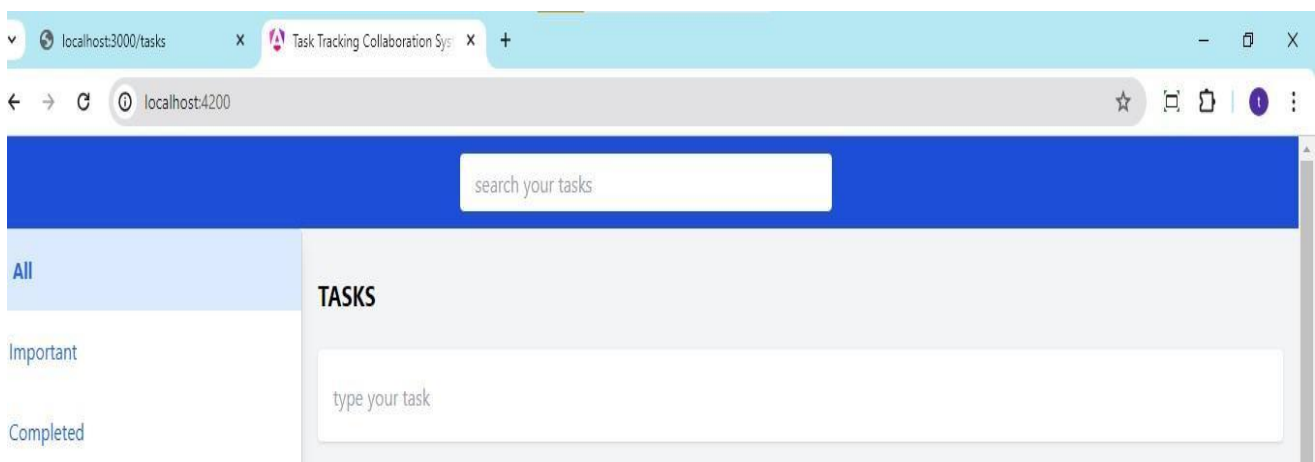


A screenshot of a login form titled "Login". It contains two input fields: "Username" and "Password". Below the "Password" field are two buttons: "Login" (in blue) and "Register" (in light blue).

File Upload Page

Description:

Admin can Upload Task details in this page



A screenshot of a web application titled "Task Tracking Collaboration System". The browser address bar shows "localhost:4200". The application has a blue header with a search bar labeled "search your tasks". On the left, there is a sidebar with three filters: "All" (selected), "Important", and "Completed". The main content area is titled "TASKS" and contains a large text input field labeled "type your task".

Manage User Page

Description:

Admin can Manage User details in this page

[Home](#) [Users](#) [Logout](#)

Users

Add User

First Name	Last Name	Username		
M.	Kavi	kavinila12@gmail.com	Edit	Delete
K.	Nila	nilavani02@gmail.com	Edit	Delete
K.Nila	Nila	nilavani01@gmail.com	Edit	Delete
M.Nila	Nila	nilavani04@gmail.com	Edit	Delete
M.Kaviya	Kaviya	kaviyakaviya02@gmail.com	Edit	Delete
M.Vani	Vani	vanivani01@gamil.com	Edit	Delete
J.Venila	Venila	venilaraj09@gamil.com	Edit	Delete
M.Mathi	Mathi	mathinila08@gmail.com	Edit	Delete

User Registration

Description:

User can Register our details in this page

Register

First Name

Last Name

Username

Password

Register

Cancel

Login Page

Description:

User can login using mail-id and password

Login

Username

Password

Login

[Register](#)

Task Page

Description:

User can View All Tasks & Status in this page

←

→

↺

🔄

🔍 localhost:4200

☆

🖨

🔧

👤

⋮

search your tasks

All

Important

Completed

TASKS

type your task

Tasks

Learn Angular Basics

☆

✔

Calculator App With Angular

☆

✔

Build a Tic-Tac-Toe Game

★

✔

Ecommerce App with Angular

★

✔

Quiz Application

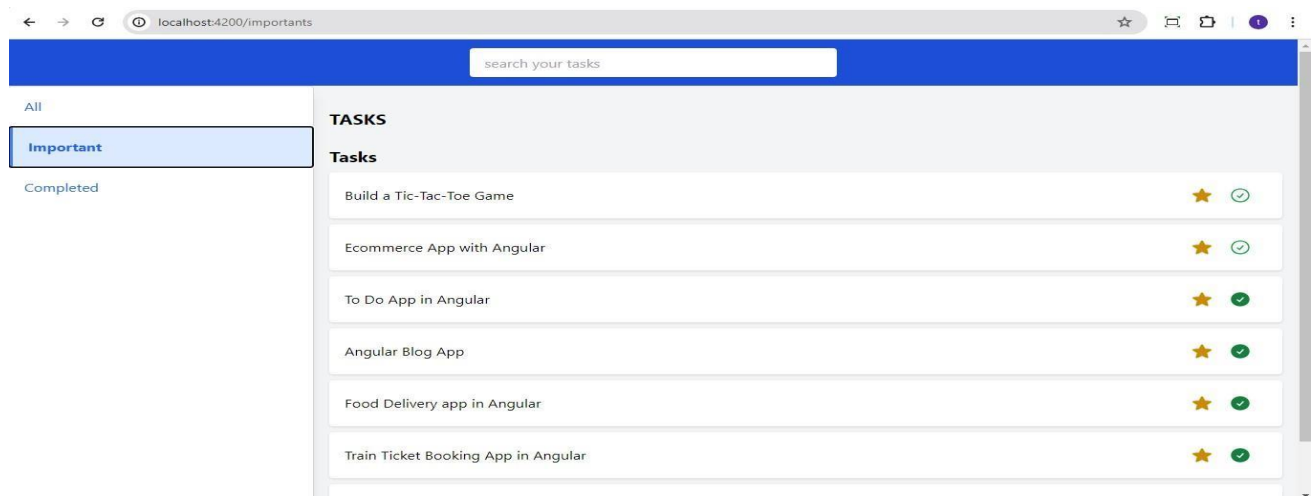
☆

✔

Task Page

Description:

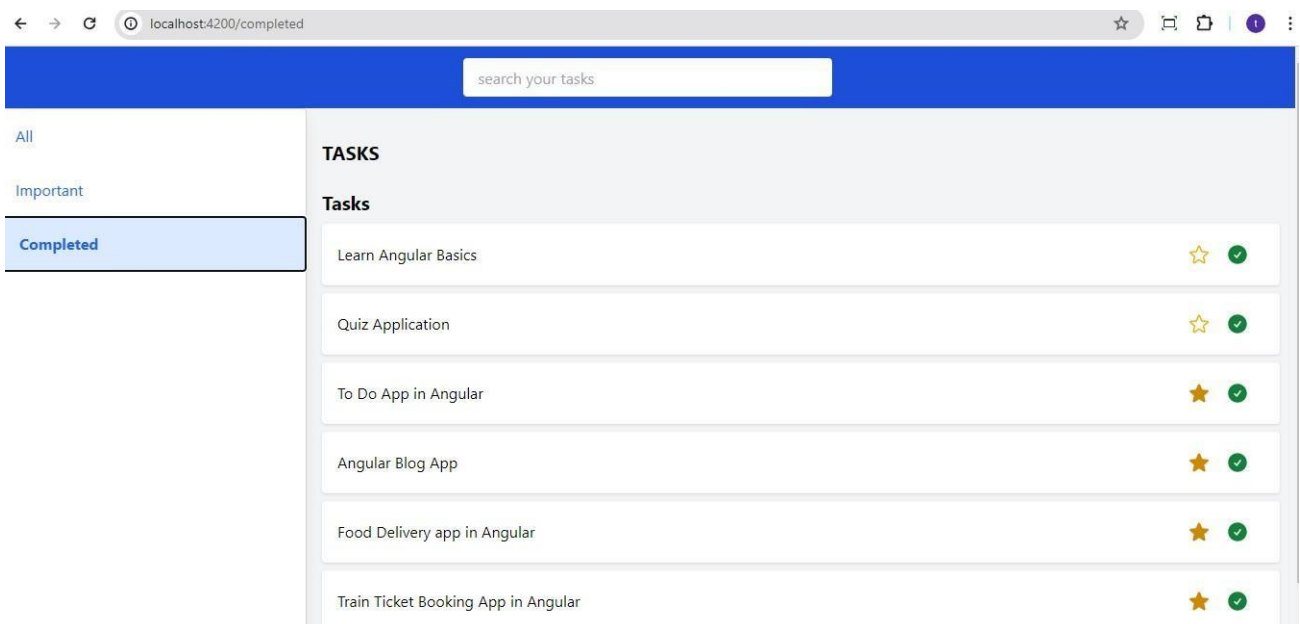
User can View Important Tasks in this page



Task Page

Description:

User can View Completed Tasks in this page



BIBLIOGRAPHY

BOOK REFERENCE

- Shyam Seshadri “Angular: Up and Running: Learning Angular, Step by Step” First published 2018. ‘O’ Really.
- Doguhan Uluca “Angular for Enterprise Ready Web Applications” First published Packet Publishing.

WEBSITES REFERENCE

- <https://angular.io/guide/http>
- <https://www.positronx.io>
- <http://www.bezkoder.com>

LIST OF FIGURES

FIG NO	DESCRIPTION	PAGE NO
5.4.1	SYSTEM FLOWDIAGRAM	11
5.4.2	DATA FLOW DIAGRAM	12
5.4.3	USE CASE DIAGRAM	14

LIST OF ABBREVIATIONS

S.NO	DESCRIPTION	ABBREVIATION
1	DFD	Data Flow Diagram
2	HTML	Hypertext Markup Language
3	CSS	Cascading Style Sheets
4	TS	Typescript

LIST OF TABLES

TABLE.NO	DESCRIPTION	PAGE NO
5.5.1	ADMIN_DETAILS	15
5.5.2	EMPLOYEE_DETAILS	15