DETAILED DESIGN

Project code:	**TDL**
Project Title:	**TO DO LIST**

Prepared by/Date **Reviewed by/Date** **Approved by/Date**

TABLE OF CONTENTS

- ## Introduction:
  - **Purpose:** The purpose of the project is to develop a digital to-do list application that allows users to manage and organize their tasks efficiently.
  - **Scope:** The application will provide features for creating, editing, and deleting tasks, updating,setting due dates and priorities, and organizing tasks into categories or projects.

- ## Assumptions and Dependencies:
  - The application will require an internet connection to function.
  - Users will have access to compatible devices (computers, smartphones, tablets) to use the application.

- ## Functional Requirements:

  - **User Registration and Authentication:**
    > The application should allow users to create an account or log in with existing credentials.
    > User authentication should be implemented to ensure secure access to the application.

  - **Task Management:**
    > Users can be able to create a new task by providing a Task name,start date,end date,status.
    > Users can be able to update task details such as start date,due date,status.
    > Users can be able to mark tasks as Not Started,completed.
    > Users can be able to delete the Task.
    > Users can be able to search for specific tasks using Task name.
    >Users can also be able to search Task by Date.

  - **Task status Tracking:**
    > The application shall allow users to mark tasks as completed ,Not started,Yet to start.
    > The application shall provide visual indicators to differentiate between completed,Not started and Yet to start.

  - **Reminders and Notifications:**
    > Users can receive notification about the Task.Notification can be sent through email as soon as the task is added by the user.

  - **User Interface:**
    > The application should have a user-friendly interface.
    > It should provide clear navigation and easy access to various features.
    > The design should be responsive and adaptable to different screen sizes and devices.

❖ Non-Functional Requirements:

- **Performance:** The application should be responsive and provide quick response times.
- **Security:** User data should be securely stored and transmitted.
- **Reliability:** The application should be reliable and available for use at all times.
- **Scalability:** The application should be able to handle a growing number of users and tasks.
- **Compatibility:** The application should be compatible with major web browsers and mobile platforms.

❖ External Interface Requirements:

- **USER INTERFACES:**
  - **>** Front-end software:HTML+CSS,BootStrap,JS

- **HARDWARE INTERFACES:**
  - **>** Windows.
  - **>** A browser that supports HTML

- **SOFTWARE INTERFACES:**
  - **>** JDK-1.8
  - **>** Apache Tomcat-9.0
  - **>** MySQL-mysql-connector-java-8.0.30.jar
  - **>** J2EE
  - **>** Back-end software:SQL

- **TECHNOLOGY USED:**
  - **>** JSP
- **PROGRAMMING LANGUAGES:**
  - **>**Java

❖ Constraints:

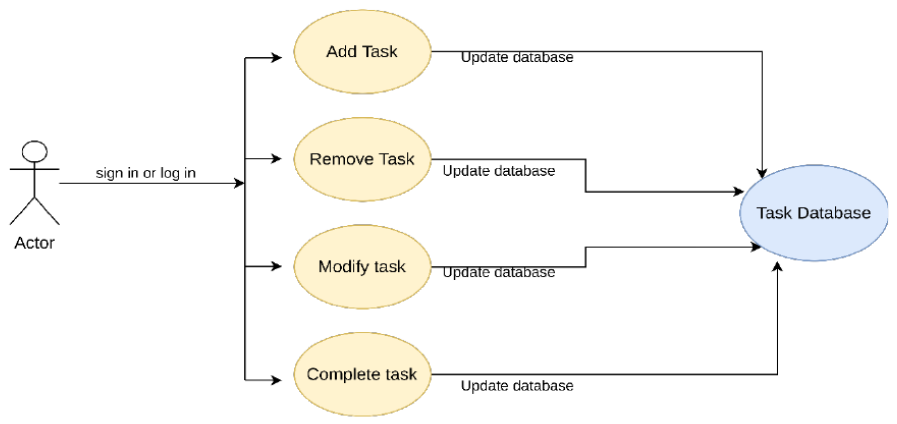
- The project should be developed within a specific time frame and budget.
- The application should utilize suitable technologies and frameworks that meet the project requirements.
- The project team should adhere to relevant coding standards and best practices.

5. HIGH LEVEL DESIGN:

This section describes the high level design diagrams. Use case diagram with Use Case definition, Sequence Diagram and Class Diagram which provides a visual representation of the requirements, logical flow and their class representations.

5.1Use Case Diagram:

The requirements of a system can be represented using a use case model in the Use Case Diagram. The use case diagram for the actors of this case study is given as below.



Login:

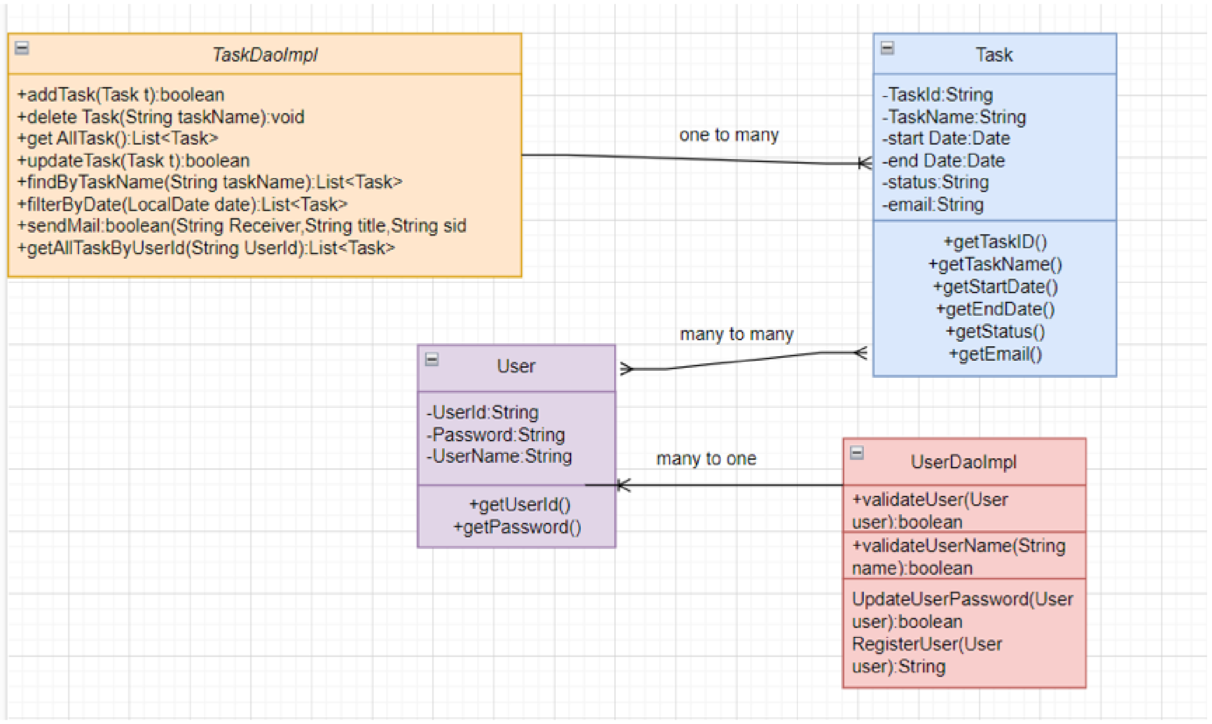
USE CASE #	Login
Goal	All users logging into the system should be authenticated using a unique login-id and password
Success End Condition	If the user enters the right credentials, then redirect to the Home page.
Failed End Condition	The end user is redirected to an Error Page and displays both passwords not matched or Invalid userId/Password.

Trigger	**Login button**	
Description	**Step**	**Action**
	1	Enter Login credentials (id & password)
	2	Click on Login button
	3	If id & password is Success, then identify user type Display appropriate(Admin/Tech/HR) home page
	step	**Branching Action**
	1	If 'Userid' is not existing then return with requesting for Registration
	2	If password is not matching return with suitable error message say both passwords not matched.
Assumptions	Admin/Tech/HR login credentials are available in the Database and others are already registered with their credentials	

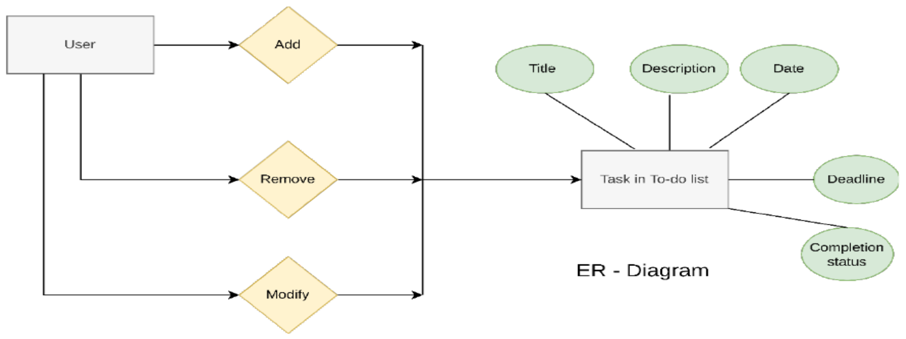
❖ **Class Diagram, an ER Diagram, and the corresponding database table names and relationships for a to-do list** :

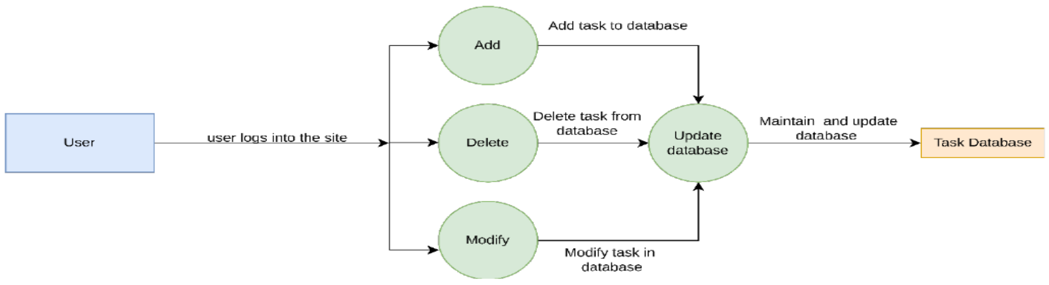
- In this representation, the class diagram shows the class structure in Java, where the data types are represented as Java data types.
- The ER diagram depicts the relationships between the entities.
- The database table names are represented with the corresponding columns and their data types.

5.2 Class Diagram:

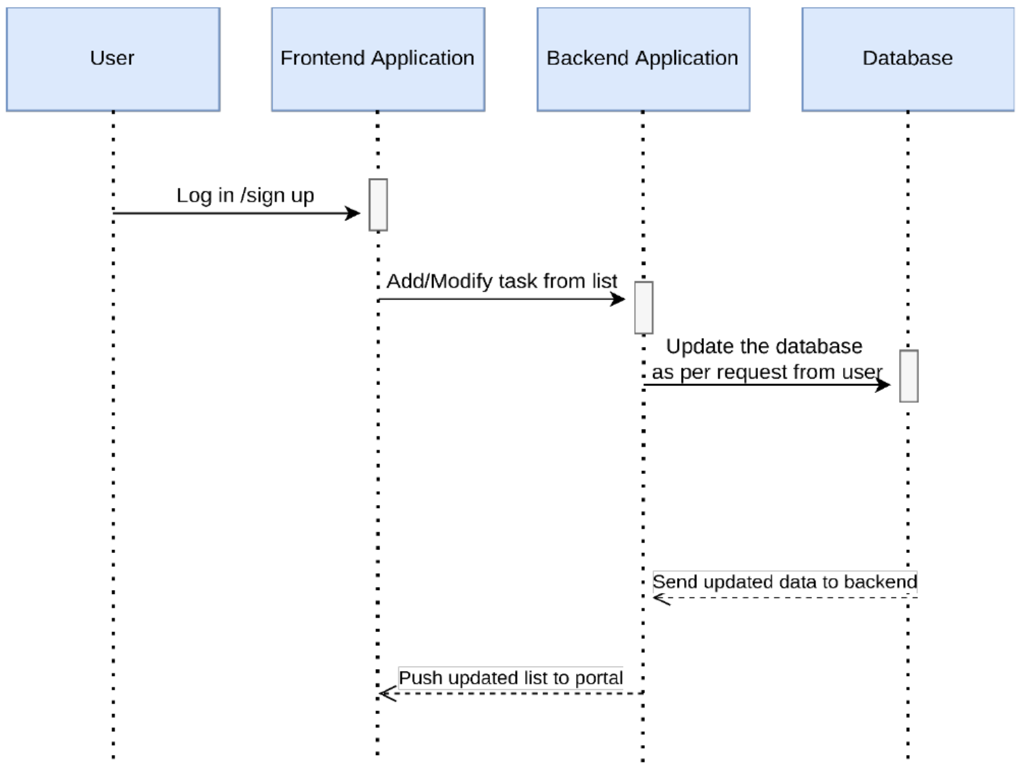

TaskDaoImpl
+addTask(Task t):boolean
+delete Task(String taskName):void
+get AllTask():List<Task>
+updateTask(Task t):boolean
+findByTaskName(String taskName):List<Task>
+filterByDate(LocalDate date):List<Task>
+sendMail:boolean(String Receiver,String title,String sid
+getAllTaskByUserId(String UserId):List<Task>
Task
-TaskId:String
-TaskName:String
-start Date:Date
-end Date:Date
-status:String
-email:String
+getTaskID()
+getTaskName()
+getStartDate()
+getEndDate()
+getStatus()
+getEmail()
one to many
many to many
User
-UserId:String
-Password:String
-UserName:String
+getUserId()
+getPassword()
many to one
UserDaoImpl
+validateUser(User user):boolean
+validateUserName(String name):boolean
UpdateUserPassword(User user):boolean
RegisterUser(User user):String

5.3 ER Diagram & Data Flow Diagram:





5.4 Sequence Diagram:


User
Frontend Application
Backend Application
Database
Log in /sign up
Add/Modify task from list
Update the database as per request from user
Send updated data to backend
Push updated list to portal

6. PACKAGES / CLASSES / INTERFACE:

This section provides a brief outlook on the packaging hierarchy along with the respective classes to be used for the implementation. The 4 packages mentioned below are for both GUI and Web Application.

Packages	
Package	**Description**
com.jdbc.task.entity	This package contains all the bean classes
com.jdbc.task.dao	This package contains all the DAO functionality classes

6.1 Package com.jdbc.task.entity:

Class Name	Attributes	Data Type
Task	taskID	String
	taskName	String
	startDate	LocalDate
	endDate	LocalDate
	status	String
	email	String
User	UserId	String
	Password	String
	Username	String

6.2 .a. Package com.jdbc.dao:

Interface Name	Description
ITaskDao	Methods
	boolean addTask(Task t)
	void deleteTask(String taskName)
	boolean updateTask(Task t
	public List<Task> findByTaskName(String taskName,String userid)
	public List<Task> FilterByDate(LocalDate date,String userid)
	List<Task>getAllTask()
	void sendEmailNotification(String emailId,String taskName,LocalDate startDate,LocalDate endDate)
	List<Task>getAllTaskByUserID(String Userid)

If required, additional methods can also be created.

b. Package com.jdbc.dao:

Class Name	Methods
MyConnection	private MyConnection()
	public static Connection getConnection()
TaskDaoImpl	public TaskDaoImpl()
	public boolean addTask(Task t)
	public void deleteTask(String taskName)
	public boolean updateTask(Task t)
	public List<Task> findByTaskName(String taskName,String userid)
	public List<Task> FilterByDate(LocalDate date,String userid)
	public List<Task>getAllTask()
	public void sendEmailNotification(String emailId,String taskName,LocalDate startDate,LocalDate endDate)
	public List<Task> getAllTaskByUserID(String email)
UserDaoImpl	public UserDaoImpl()
	public String validateUser(User user)
	public Boolean ValidateUserName(String name)
	public Boolean updateUserPassword(User user)
	public String registerUser(User user)

7.UI TEMPLATES:

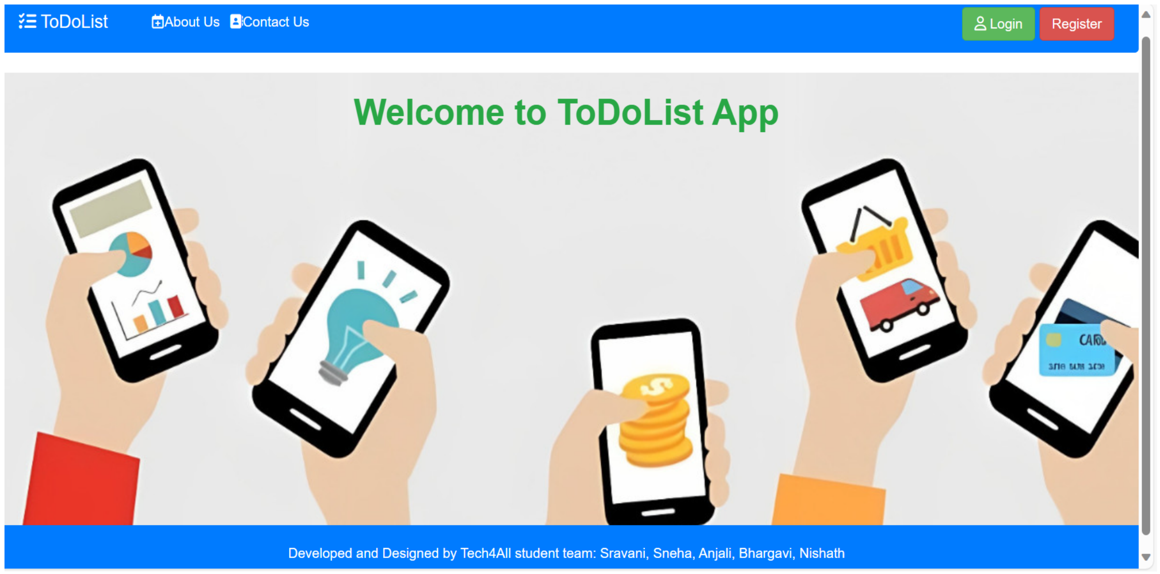
7.1 UI Principle:

The UI [Presentation Layer] should be designed with the below mentioned principles which helps easy interaction by the user to the application.

7.2 UI controls and Usage Principle

UI Type	Controls	Description
Direct Entry	Text Box, Text Area	Any input that cannot be predicted and needs the user to key in. e.g Name, Address, contact no etc.
Static Selection	Option Button, Check Box, Drop Down	Should be used where the input can be predefined. e.g gender, month [Jan – Dec] etc. If number of items is more, drop down is preferred.
Dynamic Selection	Drop Down	The items for the drop down should be retrieved from a stored Data. e.g Displaying Districts in a drop down from places table.
Automation	Label Text Field [Read Only]	Data's that are calculative or an output of a function. e.g : Displaying system date, showing total amount etc
Decision Control	Button	Operations like submit, save, clear should be executed only upon clicking respective buttons.

7.3 UI Templates:


ToDoList
About Us
Contact Us
Login
Register
Welcome to ToDoList App
Developed and Designed by Tech4All student team: Sravani, Sneha, Anjali, Bhargavi, Nishath

section contains the design template for the website home page [Fig. 1] that will be displayed at the time of opening this web application .

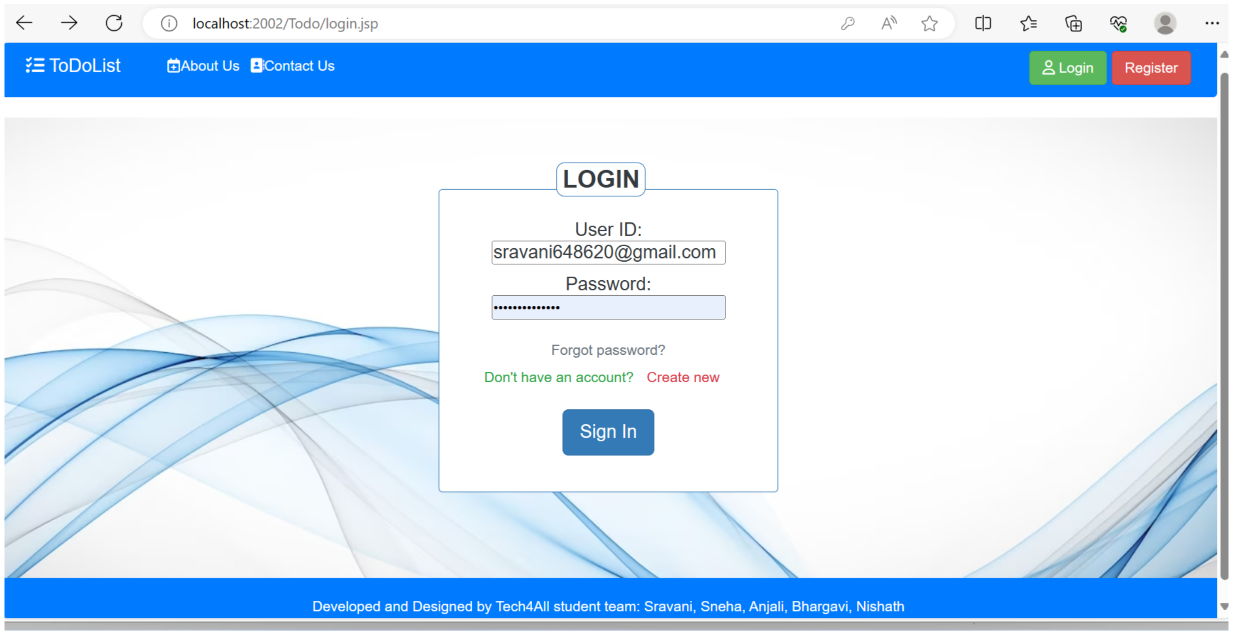
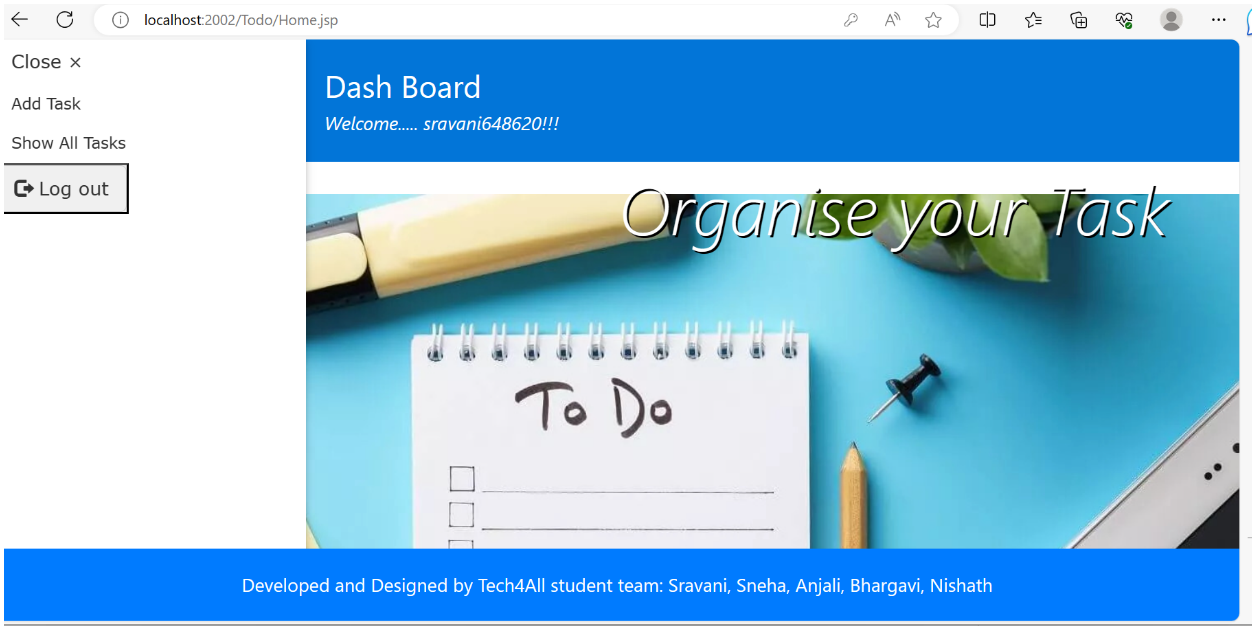


Fig. 2 - Main Page [First Page to open]

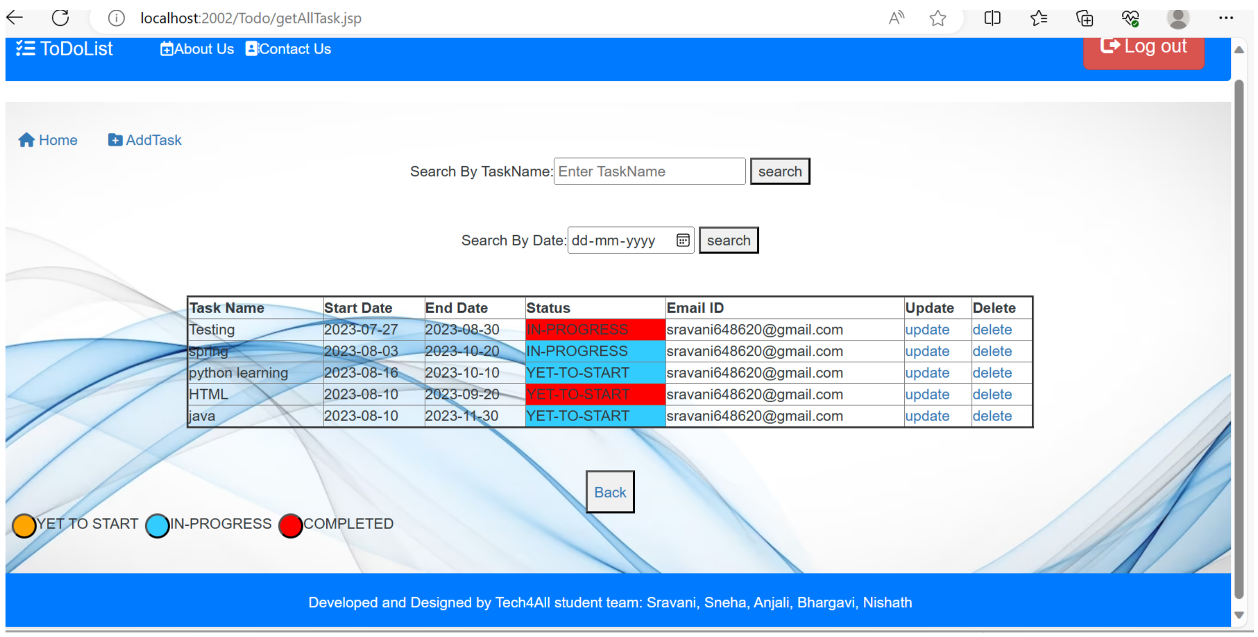


Dash Board

Welcome..... sravani648620!!!

Developed and Designed by Tech4All student team: Sravani, Sneha, Anjali, Bhargavi, Nishath

Fig. 3 - Home Page/Dash Board for User



Search By TaskName: Enter TaskName [search]

Search By Date: dd-mm-yyyy [search]

Task Name	Start Date	End Date	Status	Email ID	Update	Delete
Testing	2023-07-27	2023-08-30	IN-PROGRESS	sravani648620@gmail.com	update	delete
spring	2023-08-03	2023-10-20	IN-PROGRESS	sravani648620@gmail.com	update	delete
python learning	2023-08-16	2023-10-10	YET-TO-START	sravani648620@gmail.com	update	delete
HTML	2023-08-10	2023-09-20	YET-TO-START	sravani648620@gmail.com	update	delete
java	2023-08-10	2023-11-30	YET-TO-START	sravani648620@gmail.com	update	delete

● YET TO START ● IN-PROGRESS ● COMPLETED

Developed and Designed by Tech4All student team: Sravani, Sneha, Anjali, Bhargavi, Nishath

Fig. 4– View Screen with update,Delete and for finding a task by task name/date Functionality

7.4. Critical Functions :

login(), addTask(), deleteTask(),getAllTask(),UpdateTask().

7.5. Limitations:

- Adding Task details will be performed by User.
- Candidate should login to view any kind of information.

8. APPENDIX:

8.1.Table:User

Field Name	Data Type	Description
UserId	varchar(30)	Not Null
Password	varchar(15)	Not Null
Username	varchar(20)	Not Null

8.2.Table:Task

Field Name	Data Type	Description
taskId	int	Primary key ,auto_increment
taskName	varchar(20)	Not Null
startdate	date	Not Null
endDate	date	Not Null
status	varchar(10)	Not Null
email	varchar(20)	Not Null

❖ **Relationships:**

- Multiple users can have multiple Tasks (many-to-many relationship between users and tasks)
-TaskDaoImpl class manages the creation,modification,and deletion of Tasks.
-One-to-many relationship between TaskDaoImpl and Task.
-Many-to-one relationship between User and UserDaoImpl.

THE END