INSTITUTIONAL TRAINING

PROJECT REPORT

ON

TASKS MANAGER

Submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE & ENGINEERING

Department of Computer Science & Engineering
Chandigarh University, Gharuan

Submitted to:

Er. **Reetu Kumari**

Er. **Vishal Kumar**

Submitted by:

AARYA RAJOJU (UID - **20BCS769**)

Table Of Contents

CERTIFICATE

This is to certify that the work embodied in this Project Report entitled "**TASKS MANAGER**", being submitted by "**20BCS7669**" during the institutional training (2nd semester) for partial fulfillment of the requirement for the degree of "**Bachelor of Engineering in Computer Science & Engineering**" discipline in "**Chandigarh University**" during the training period June-July 2021 is a record of bona fide piece of work, carried out by the student under my supervision and guidance in the "**Department of Computer Science & Engineering**", Chandigarh University.

DECLARATION

I, a student of Bachelor of Engineering in Computer Science & Engineering, institutional training (2nd semester), session: June-July 2021, Chandigarh University, hereby declare that the work presented in this Project Report entitled **"TASKS MANAGER"** is the outcome of my own work, it is bona fide and correct to the best of my knowledge and this work has been carried out taking care of Engineering Ethics.
The work presented does not infringe any patent work and has not been submitted to any other university or anywhere else for the award of any degree or any professional diploma.

STUDENT NAME AND SIGNATURE

Aarya Rajoju

Aarya Rajoju (UID: 20BCS7669)

APPROVED BY:
Er. Reetu Kumari Ma'am

GUIDED BY:
Er. Vishal Kumar Sir

ACKNOWLEDGEMENT

We would like to express our deep and sincere gratitude to our Project In-charge Er. Reetu Kumari Ma'am & Er. Vishal Kumar Sir for giving me the opportunity to do the project and providing valuable guidance throughout this research. Their dynamism, vision, and exquisite efforts have deeply inspired me. They taught us the methodology to carry out the research and to present the research work as clearly as possible. It was a great privilege for me to study and work under their guidance

ABSTRACT

A tasks management app is a tool for keeping a track of all the tasks to be done.

A task management tool is used by an individual, team, or organization to help complete projects more efficiently by organizing and prioritizing related tasks.

My aim was to create a very simple Tasks Manager web-app, that would be highly reliable, scalable, and very easy-to-use

CHAPTER 1
INTRODUCTION

1.1 Aim & Objectives

- Create an easy-to-use interface for a To-Do List app

- Have high reliability
    - use Node.JS for the local server for its reliability
    - use PostgreSQL for the DBMS for its reliability

- Be highly scalable and expandable
    - Express.JS is a highly scalable framework for web-apps
    - build a Rest API so that the same back-end can be extended to be used with a mobile app, desktop app, or any other application

- Make it easy to add more functionalities in the future, if needed
    - the HTML/CSS and JS code is really clean and straight-forward so that additional features can be easily added

1.2 Theoretical Explanation

This project is designed to be used for managing a ToDo List.

The user can see all the tasks, change the progress status of the tasks, delete old tasks, and add new ones.

The front-end webpage has a very simple interface, for the user to perform actions to the tasks list.
All of the user actions are communicated to the back-end server, through API calls.
When the back-end receives an API call for a particular action, it routes to the respective function to perform the crud-operation on the DB.
Once the curd-oration is successful, the server sends a response to the webpage, about the completion of the task, and the webpage gets updated accordingly.

As soon as the webpage first loads up, it sends a GET HTTP request to get all the tasks.
And, then, based on the user action, it sends POST (add), PUT (update), and DELETE HTTP requests, to perform the respective task on the database.

1.3 Technologies Used

- **Node.JS**
  - Node.js is an open-source, cross-platform, back-end JavaScript runtime environment that runs on Chrome's V8 engine and executes JavaScript code outside a web browser.
  - Node.js lets developers use JavaScript to write command-line tools and for server-side scripting — running scripts server-side to produce dynamic web page content before the page is sent to the user's web.

- **Express.JS**
  - Express.js, or simply Express, is a back-end web application framework for Node.js, released as free and open-source software under the MIT License.
  - It is designed for building web applications and APIs. It has been called the de facto standard server framework for Node.js.

- **PostgreSQL**
  - PostgreSQL, also known as Postgres, is a powerful, open-source object-relational database system, that has a strong reputation for its reliability, flexibility, and support of open technical standards
  - Unlike other RDMBS, PostgreSQL supports both relational and non-relational data types. This makes it one of the most compliant, stable, and mature relational databases available today.

Programming/Scripting Languages Used:

- **JavaScript**
  - JavaScript, often abbreviated as JS, is a programming language that conforms to the ECMAScript specification.
  - JavaScript is high-level, often just-in-time compiled, and multi-paradigm.
  - It was originally designed as a scripting language for websites but became widely adopted as a general-purpose programming language.

- **HTML/CSS**
  - The HyperText Markup Language or HTML is the standard markup language for documents designed to be displayed in a web browser.
  - HTML can be assisted by technologies such as Cascading Style Sheets and scripting languages such as JavaScript.
  - Cascading Style Sheets or CSS is a style sheet language used for describing the presentation of a document written in a markup language such as HTML.

Additional Tools Used:

- **JetBrains WebStorm**
  - An IDE for JavaScript, and making Web-based Apps
- **PostMan**
  - A platform for API development and testing
- **Nodemon**
  - A Node.JS utility tool for active development
- **GitHub**
  - A platform for Code-Hosting and Version Control

1.4 Brief Explanation of the Working

- **DB:**
    - Postgres was set up with a new database named `todo`, and one table named `tasks`
    - in the tasks table, there are three columns - `id` (primary key), `taskname`, and `status`
    - the crud operations are handled through node.js, using the `pg` module

- **Rest API:**
    - an express app is set up to listen on the port 3000
    - `queries.js` has all the SQL queries, required for the crud operations, which are written using the `pg` node module
    - the `api.js` handles the express server and also routes the HTTP request to the respective function in `queries.js`
    - the HTTP requests are sent by the client-side js, on button presses, or text filed changes, using event listeners

- **Website:**
    - served as a static webpage, by express.js
    - the HTML file has one div for the entire body, all the elements are added to the DOM, using the `script.js` file
    - the `script.js` file also has even listeners on every button and text inputs, which send the HTTP request to the Rest API, to communicate with the DB.
    - when the page first loads, it gets all the tasks from the DB, by sending a `GET` HTTP request
    - then, based on the user action, `POST`, `PUT`, and `DELETE` HTTP requests are sent to update the DB

CHAPTER 2
SRS

(Software Requirement Specifications)

2.1 Scope

- This SRS document specifies requirements for a simple Tasks Management app.
- The application allows users to:
  - View all tasks
  - add new tasks
  - edit tasks
  - enter the progress of the tasks
  - delete tasks
- The application stores the data in a relational SQL database
- The webpage sends API calls to the server to perform actions on the database
- The application requires connecting to a server

2.2 Requirements

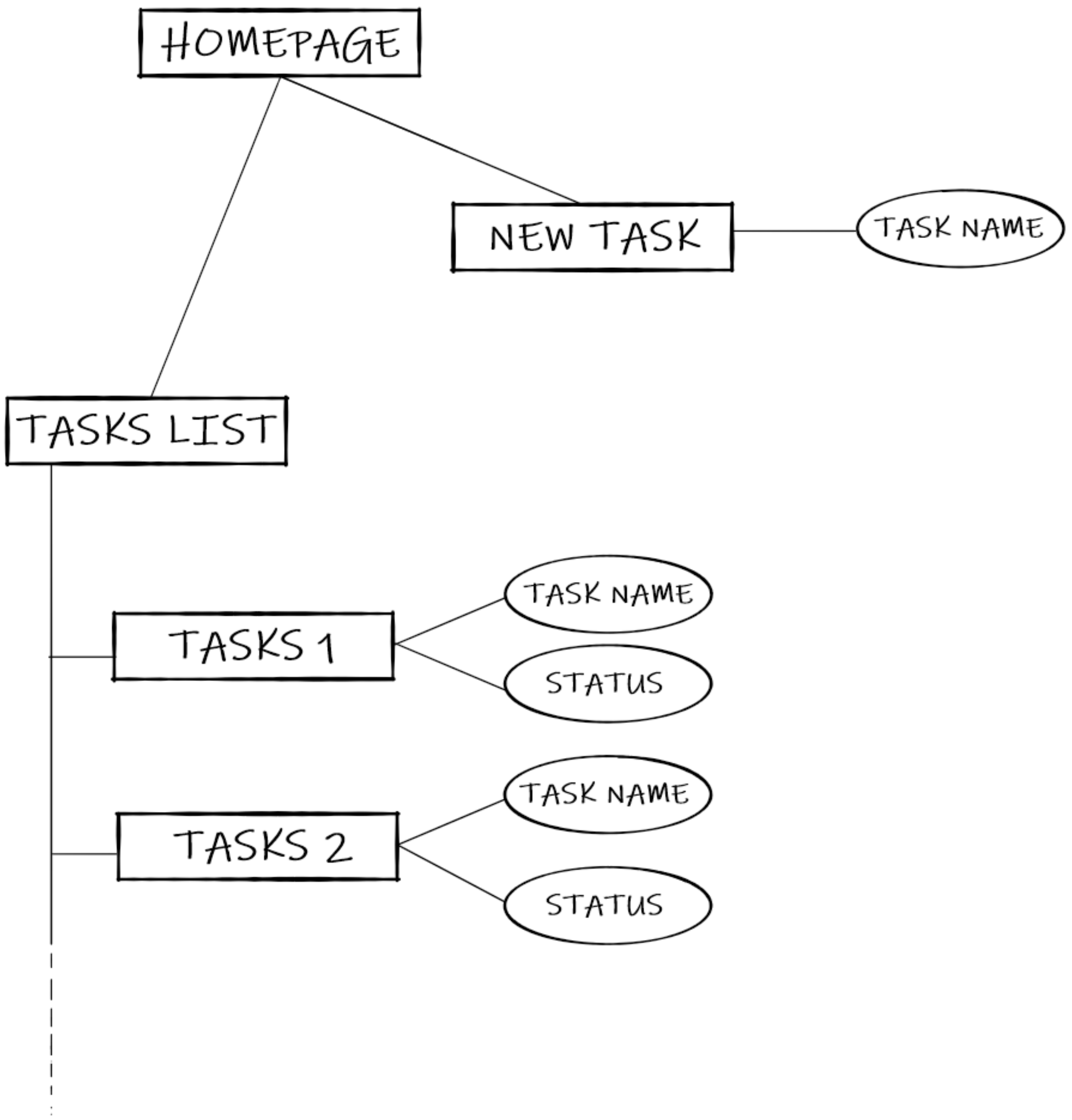
- Software Configuration :
    - Operating System: **Windows 10**
    - DBMS: **PostgreSQL**
    - LocalSerer & Back-End: **Node.JS**
    - Rest API: **Express.JS**
    - Front-End: **HTML/CSS/JS**

- Hardware Configuration :
  (hardware used for development)
    - Processor: **Intel i7-11370H (3.3 GHz)**
    - Storage: **1TB NVME-SSD**
    - RAM: **16 GB DDR4-3200**
    - Display Resolution: **1920 x 1080**

  (minimum hardware requirements)
    - Processor: **any modern multi-core CPU**
    - Storage: **at least 256 GB**
    - RAM: **at least 8 GB DDR4**
    - Display Resolution: **1920 x 1080**

CHAPTER 3
ERD

(Entity-Relationship Diagram)



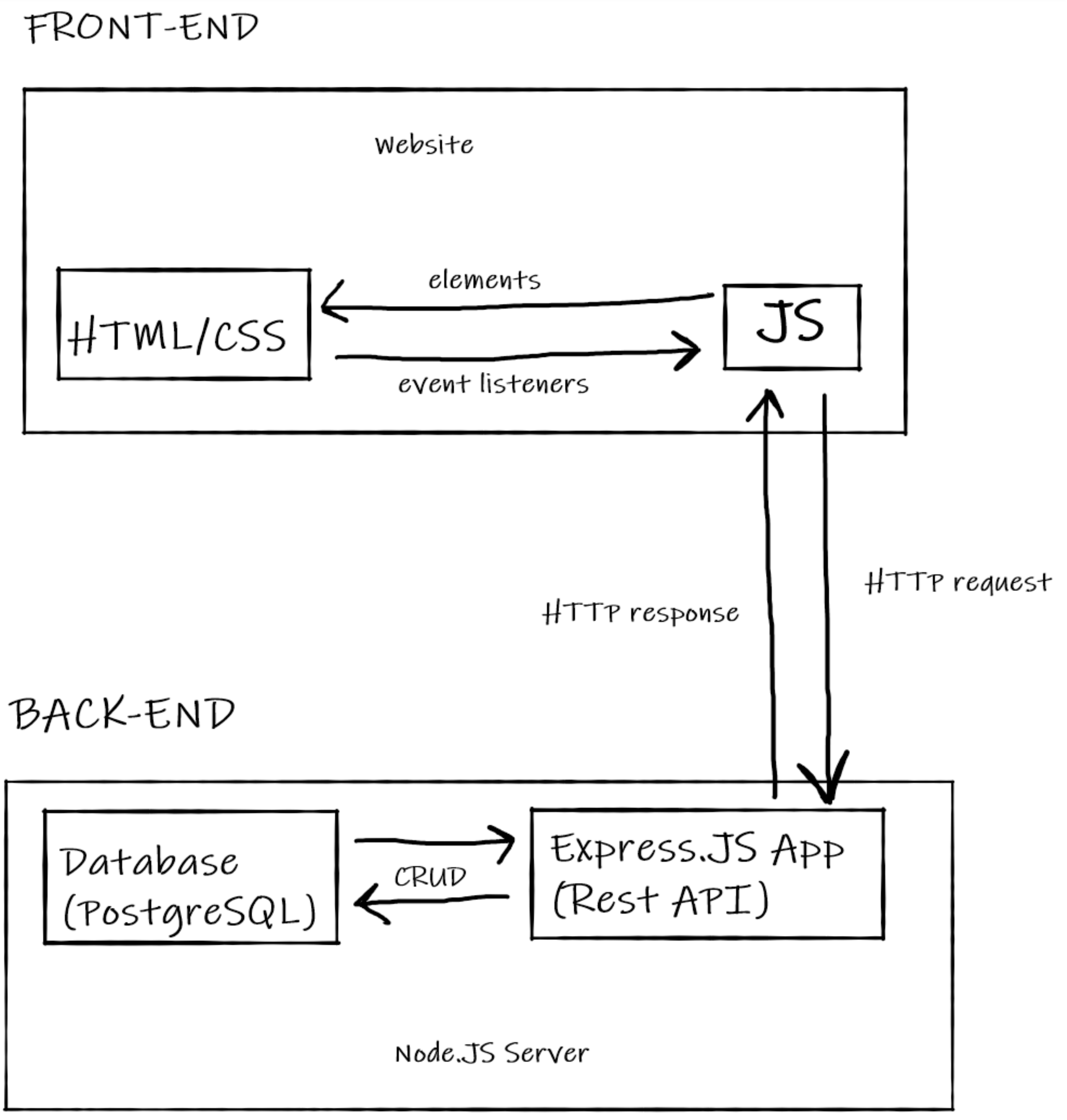
CHAPTER 4
MATERIAL AND METHODOLOGY

4.1 Methodology

- First, the database was set up with the table, and the necessary columns
- Then, the crud operations are defined in the `queries.js`, using the `pg` node module
- In the `api.js`, the express.js app is set up, with the HTTP requests to the Rest API, routing to the functions in `queries.js`
- The website is served by the express.js app, as a static webpage
- Then, the webpage is completed
- The elements on the webpage, have event listeners, which send the HTTP requests to the server, and the respective CRUD operations are carried out on the database.

4.2 Project Design

- On the front-end side, the JavaScript adds the elements to the DOM
- The elements on the DOM, have event listeners, which go to the functions defined in JS, which further send HTTP requests to the Rest API
- On the back-end side, the Rest API on Express.JS app gets the HTTP request and performs the CRUD operations on the database, and it sends the responses back to thewebsite



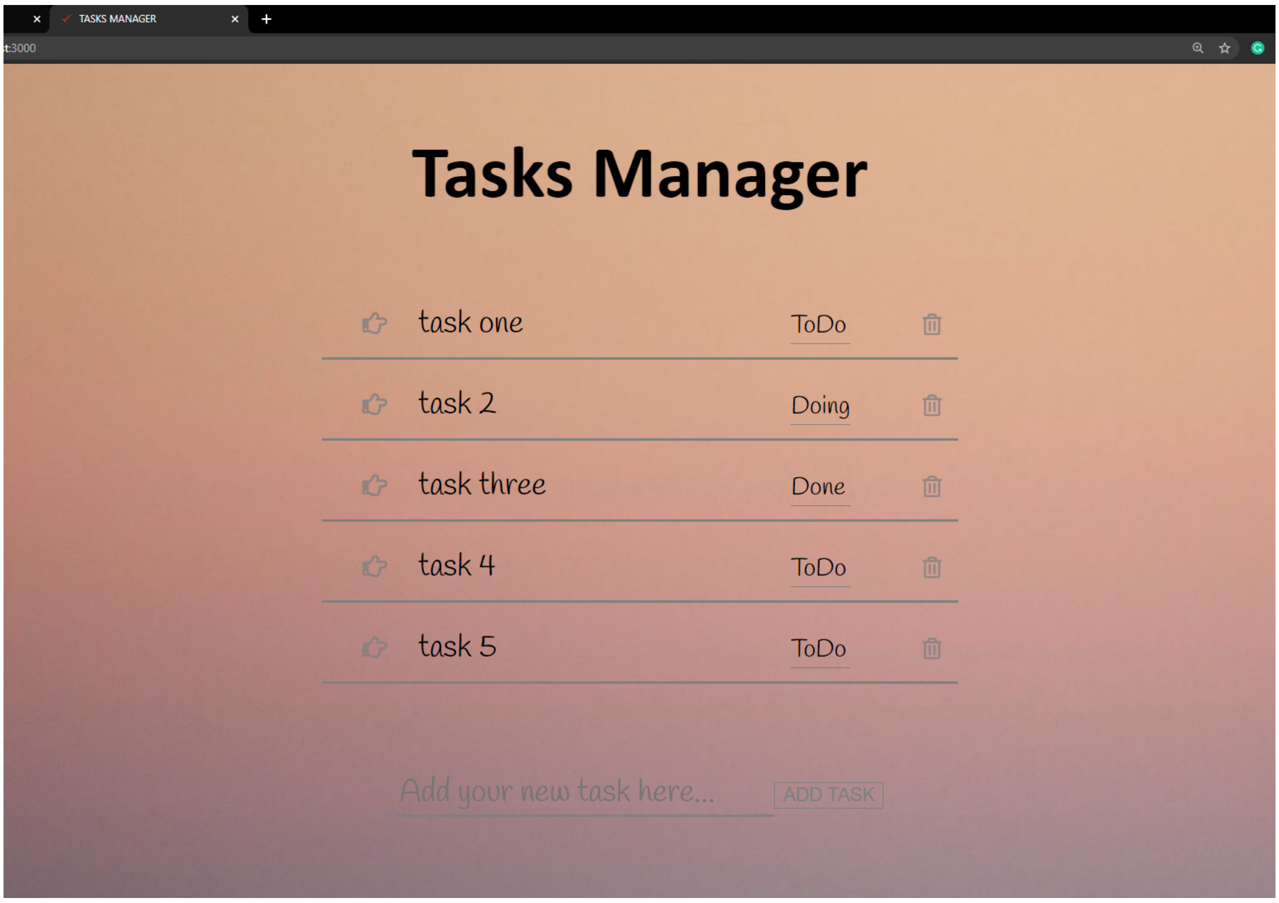
CHAPTER 5
RESULTS AND SNAPSHOTS

5.1 Results

- A very simple tasks manager app has been completed
- The webpage has a straight-forward user-friendly interface to interact with the todo list
- The application allows users to:
  - View all tasks
  - add new tasks
  - edit tasks
  - enter the progress of the tasks
  - delete tasks
- The webpage communicate with the database through API calls and HTTP requests
- The data is stored in the local server using PostgreSQL

5.2 Snapshots

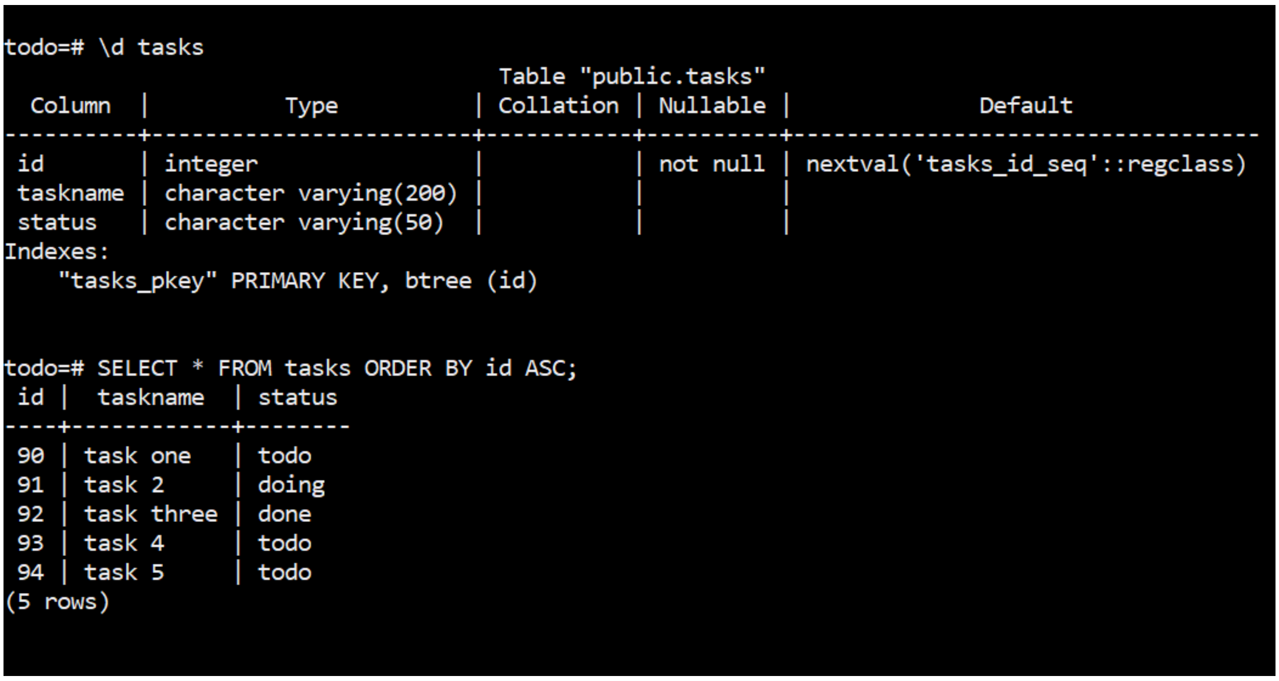
- WebPage



Tasks Manager

task one	ToDo
task 2	Doing
task three	Done
task 4	ToDo
task 5	ToDo

- DataBase



```
todo=# \d tasks
                                      Table "public.tasks"
  Column  |          Type          | Collation | Nullable |              Default
----------+------------------------+-----------+----------+-----------------------------------
 id       | integer                |           | not null | nextval('tasks_id_seq'::regclass)
 taskname | character varying(200) |           |          |
 status   | character varying(50)  |           |          |
Indexes:
    "tasks_pkey" PRIMARY KEY, btree (id)


todo=# SELECT * FROM tasks ORDER BY id ASC;
 id |  taskname  | status
----+------------+--------
 90 | task one   | todo
 91 | task 2     | doing
 92 | task three | done
 93 | task 4     | todo
 94 | task 5     | todo
(5 rows)
```

CHAPTER 6
CONCLUSION AND FUTURE SCOPE

6.1 Conclusion

- The project is designed in a way that is the most user-friendly with very simple elements on the webpage and also has a very high ease-of-use
- Adding tasks, editing task names, changing the status of tasks, and deleting tasks has been made in a way that is the most intuitive to the end-user
- The technologies used in the making of the project (node.js, express.js, PostgreSQL) were chosen so that it would be the easiest to develop and were also better than their alternatives
- Instead of using technologies from the 1900s (PHP), this project makes use of modern technologies (such as node.js)
- By building and using a Rest API, the project has become very flexible and highly scalable for the future

6.2 Future Scope

- A lot of additional functionalities can be added to this application to increase the feature-set:
    - Adding support for multiple users, by creating a login and authentication system
    - Adding `Sign in with Google` for login
    - Adding support for multiple todo lists
    - Adding support for categorizing tasks
    - Adding `due date` to the tasks
    - Adding `remainders`
    - Adding support for user-customizable background image
    - Adding productivity features, such as Pomodoro timers

- Because of how the application is built, it can be easily scaled to other platforms
    - Build a mobile app
    - Build a desktop app
    - Build a console app

DELIVERABLES

- **GitHub repo:**
  - https://github.com/aaryarajoju/ToDo-Manager

- **Project Report**
  - https://github.com/aaryarajoju/ToDo-Manager/blob/main/submissions/ProjectReport.pdf

- **Presentation**
  - https://github.com/aaryarajoju/ToDo-Manager/blob/main/submissions/Presentation.ppsx

REFERENCES

- **Expess.JS documentation:**
  - http://expressjs.com/en/5x/api.html
  - http://expressjs.com/en/guide/routing.html
- **PostgreSQL documentation:**
  - https://www.postgresql.org/docs/
- **W3Schools**
  - https://www.w3schools.com/

THE END