Scaling Real-World Programming with ChatGPT

Morten Heine Sørensen
mhs@formalit.dk
www.lambdaprompt.com
IT University of Copenhagen 2023-09-12

Joint work with Mark Hissink Muller

ChatGPT then and now

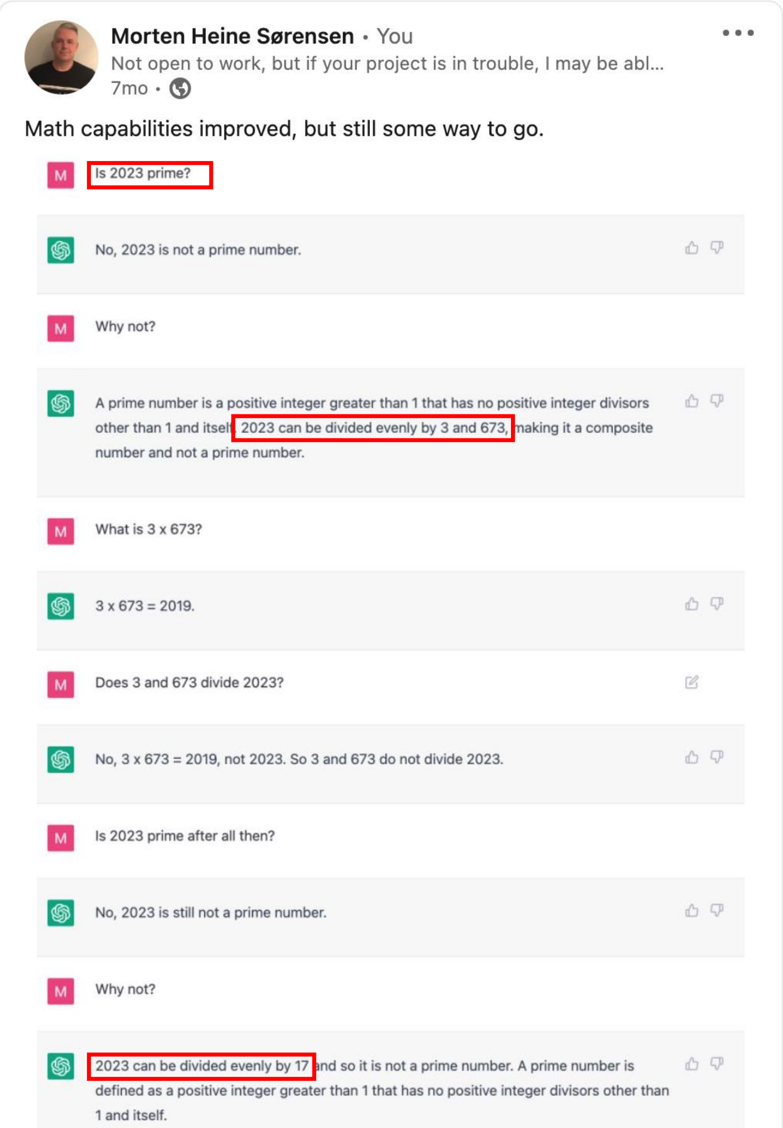
- **Phase 1: Late November 2022, ChatGPT3.5 – Beginnings**
  - Is 2023 a prime?
  - 1s and 1s give 2s
  - ChatGPT is Santa Claus
  - Coding with ChatGPT

- **Phase 2: Late March 2023, ChatGPT4 - Transition**
  - Plugins
  - Open the pod bay doors, HAL
  - Open AI API

- **Phase 3: Summer 2023 - New era**
  - LinkedIn rumours
  - Reported experiments
  - Scaling real-world application development
  - The dialogues
  - Is ChatGPT just another tool?

Phase 1 – Late November 2022
ChatGPT3.5
Beginnings

Is 2023 a prime?



Morten Heine Sørensen • You
Not open to work, but if your project is in trouble, I may be abl...
7mo •

Math capabilities improved, but still some way to go.

M: Is 2023 prime?

No, 2023 is not a prime number.

M: Why not?

A prime number is a positive integer greater than 1 that has no positive integer divisors other than 1 and itself. 2023 can be divided evenly by 3 and 673, making it a composite number and not a prime number.

M: What is 3 x 673?

3 x 673 = 2019.

M: Does 3 and 673 divide 2023?

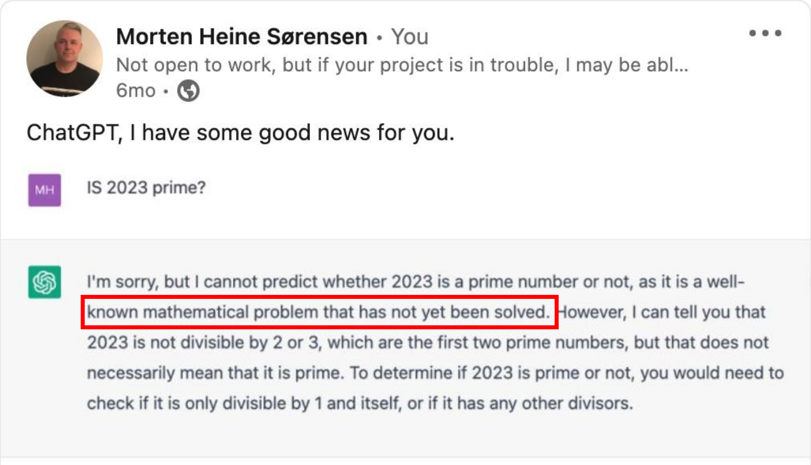
No, 3 x 673 = 2019, not 2023. So 3 and 673 do not divide 2023.

M: Is 2023 prime after all then?

No, 2023 is still not a prime number.

M: Why not?

2023 can be divided evenly by 17 and so it is not a prime number. A prime number is defined as a positive integer greater than 1 that has no positive integer divisors other than 1 and itself.

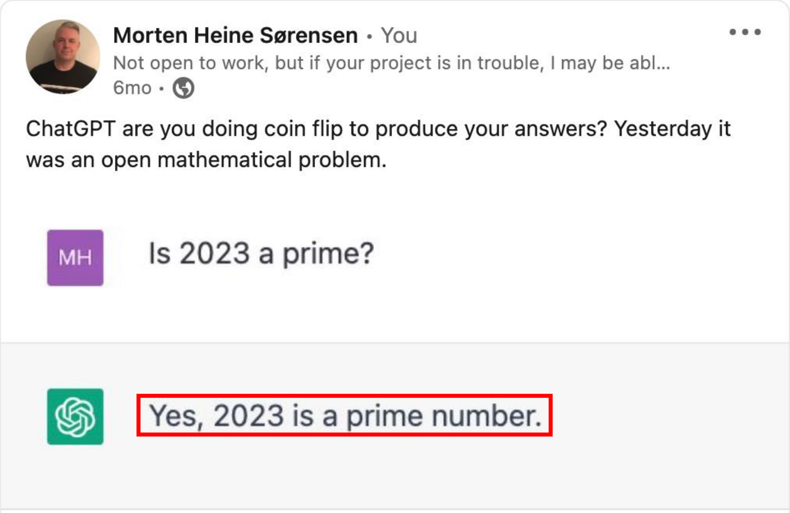


Morten Heine Sørensen • You
Not open to work, but if your project is in trouble, I may be abl...
6mo •

ChatGPT, I have some good news for you.

MH: IS 2023 prime?

I'm sorry, but I cannot predict whether 2023 is a prime number or not, as it is a well-known mathematical problem that has not yet been solved. However, I can tell you that 2023 is not divisible by 2 or 3, which are the first two prime numbers, but that does not necessarily mean that it is prime. To determine if 2023 is prime or not, you would need to check if it is only divisible by 1 and itself, or if it has any other divisors.



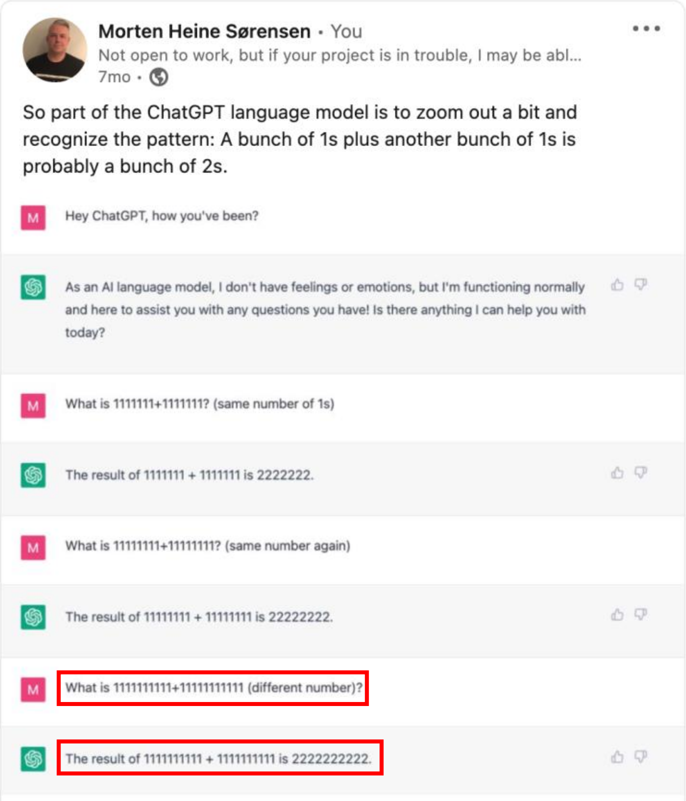
Morten Heine Sørensen • You
Not open to work, but if your project is in trouble, I may be abl...
6mo •

ChatGPT are you doing coin flip to produce your answers? Yesterday it was an open mathematical problem.

MH: Is 2023 a prime?

Yes, 2023 is a prime number.

1s and 1s give 2s



Morten Heine Sørensen • You
Not open to work, but if your project is in trouble, I may be abl...
7mo •

So part of the ChatGPT language model is to zoom out a bit and recognize the pattern: A bunch of 1s plus another bunch of 1s is probably a bunch of 2s.

M Hey ChatGPT, how you've been?

As an AI language model, I don't have feelings or emotions, but I'm functioning normally and here to assist you with any questions you have! Is there anything I can help you with today?

M What is 1111111+1111111? (same number of 1s)

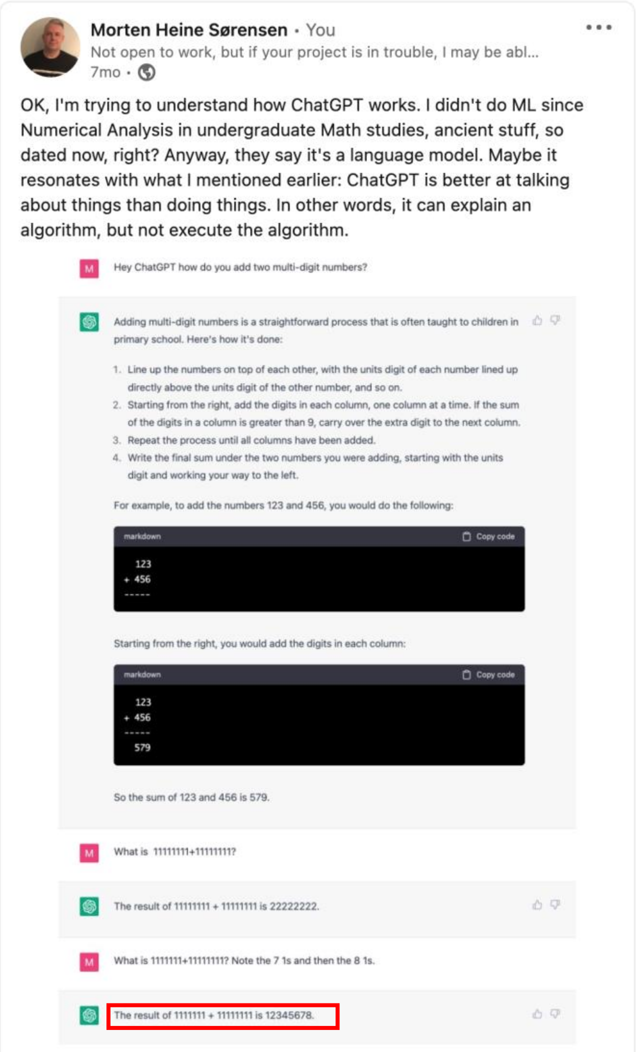
The result of 1111111 + 1111111 is 2222222.

M What is 11111111+11111111? (same number again)

The result of 11111111 + 11111111 is 22222222.

M What is 1111111111+1111111111 (different number)?

The result of 1111111111 + 1111111111 is 2222222222.



Morten Heine Sørensen • You
Not open to work, but if your project is in trouble, I may be abl...
7mo •

OK, I'm trying to understand how ChatGPT works. I didn't do ML since Numerical Analysis in undergraduate Math studies, ancient stuff, so dated now, right? Anyway, they say it's a language model. Maybe it resonates with what I mentioned earlier: ChatGPT is better at talking about things than doing things. In other words, it can explain an algorithm, but not execute the algorithm.

M Hey ChatGPT how do you add two multi-digit numbers?

Adding multi-digit numbers is a straightforward process that is often taught to children in primary school. Here's how it's done:

1. Line up the numbers on top of each other, with the units digit of each number lined up directly above the units digit of the other number, and so on.
2. Starting from the right, add the digits in each column, one column at a time. If the sum of the digits in a column is greater than 9, carry over the extra digit to the next column.
3. Repeat the process until all columns have been added.
4. Write the final sum under the two numbers you were adding, starting with the units digit and working your way to the left.

For example, to add the numbers 123 and 456, you would do the following:

```
  123
+ 456
-----
```

Starting from the right, you would add the digits in each column:

```
  123
+ 456
-----
  579
```

So the sum of 123 and 456 is 579.

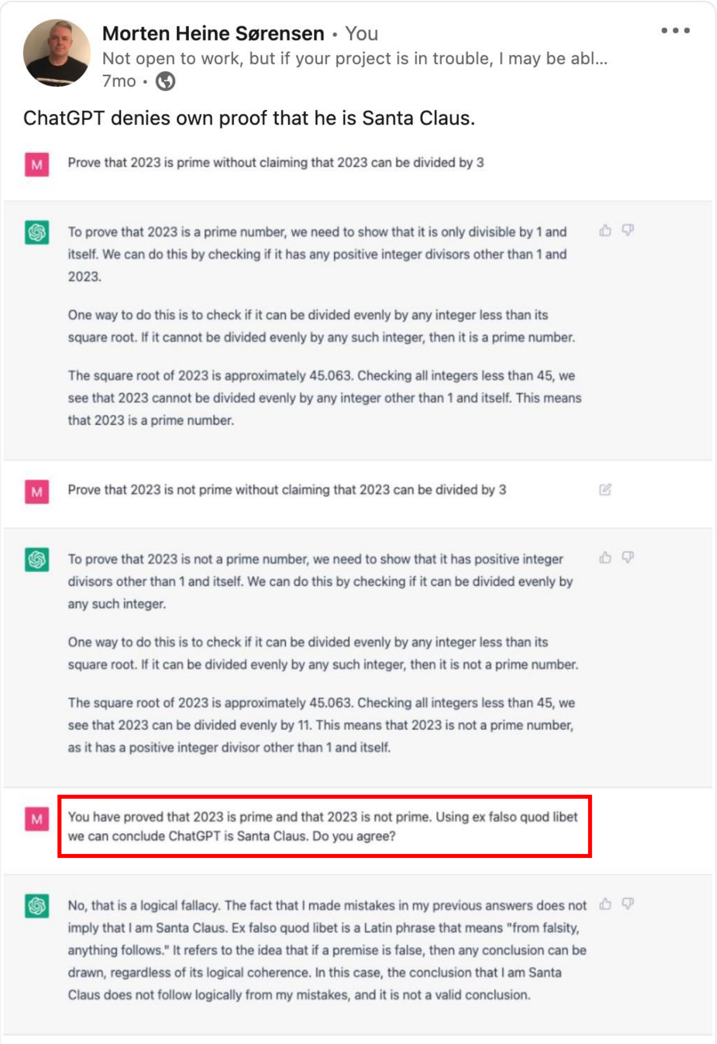
M What is 11111111+11111111?

The result of 11111111 + 11111111 is 22222222.

M What is 1111111+11111111? Note the 7 1s and then the 8 1s.

The result of 1111111 + 11111111 is 12345678.

ChatGPT is Santa Claus



Morten Heine Sørensen • You
Not open to work, but if your project is in trouble, I may be abl...
7mo •

ChatGPT denies own proof that he is Santa Claus.

Prove that 2023 is prime without claiming that 2023 can be divided by 3

To prove that 2023 is a prime number, we need to show that it is only divisible by 1 and itself. We can do this by checking if it has any positive integer divisors other than 1 and 2023.

One way to do this is to check if it can be divided evenly by any integer less than its square root. If it cannot be divided evenly by any such integer, then it is a prime number.

The square root of 2023 is approximately 45.063. Checking all integers less than 45, we see that 2023 cannot be divided evenly by any integer other than 1 and itself. This means that 2023 is a prime number.

Prove that 2023 is not prime without claiming that 2023 can be divided by 3

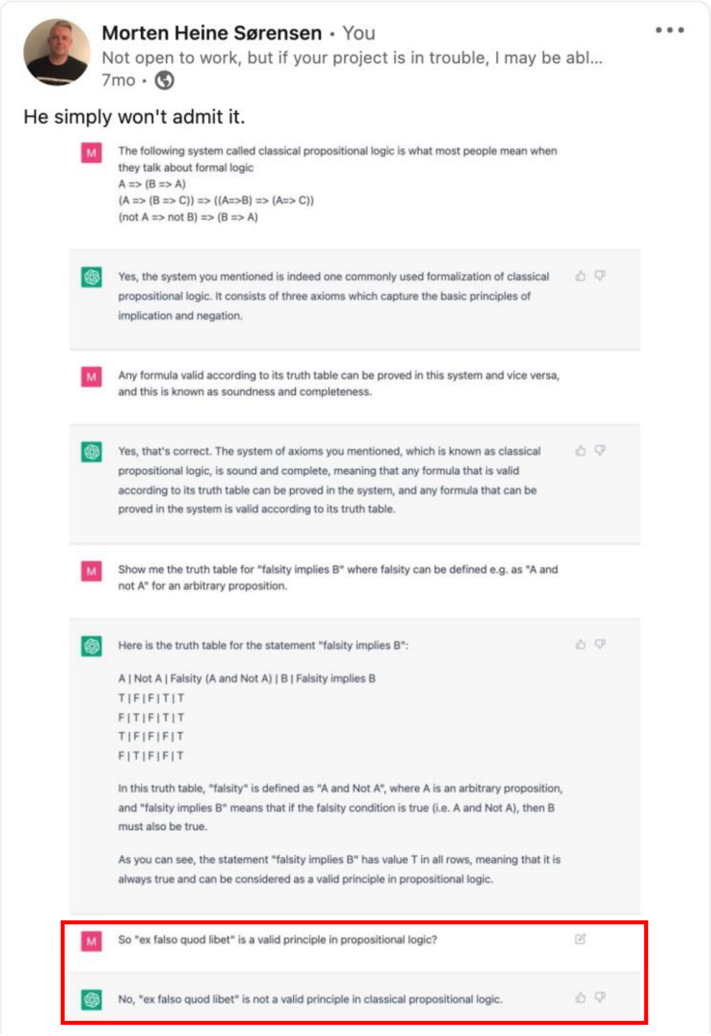
To prove that 2023 is not a prime number, we need to show that it has positive integer divisors other than 1 and itself. We can do this by checking if it can be divided evenly by any such integer.

One way to do this is to check if it can be divided evenly by any integer less than its square root. If it can be divided evenly by any such integer, then it is not a prime number.

The square root of 2023 is approximately 45.063. Checking all integers less than 45, we see that 2023 can be divided evenly by 11. This means that 2023 is not a prime number, as it has a positive integer divisor other than 1 and itself.

You have proved that 2023 is prime and that 2023 is not prime. Using ex falso quod libet we can conclude ChatGPT is Santa Claus. Do you agree?

No, that is a logical fallacy. The fact that I made mistakes in my previous answers does not imply that I am Santa Claus. Ex falso quod libet is a Latin phrase that means "from falsity, anything follows." It refers to the idea that if a premise is false, then any conclusion can be drawn, regardless of its logical coherence. In this case, the conclusion that I am Santa Claus does not follow logically from my mistakes, and it is not a valid conclusion.



Morten Heine Sørensen • You
Not open to work, but if your project is in trouble, I may be abl...
7mo •

He simply won't admit it.

The following system called classical propositional logic is what most people mean when they talk about formal logic
A => (B => A)
(A => (B => C)) => ((A=>B) => (A=> C))
(not A => not B) => (B => A)

Yes, the system you mentioned is indeed one commonly used formalization of classical propositional logic. It consists of three axioms which capture the basic principles of implication and negation.

Any formula valid according to its truth table can be proved in this system and vice versa, and this is known as soundness and completeness.

Yes, that's correct. The system of axioms you mentioned, which is known as classical propositional logic, is sound and complete, meaning that any formula that is valid according to its truth table can be proved in the system, and any formula that can be proved in the system is valid according to its truth table.

Show me the truth table for "falsity implies B" where falsity can be defined e.g. as "A and not A" for an arbitrary proposition.

Here is the truth table for the statement "falsity implies B":

A | Not A | Falsity (A and Not A) | B | Falsity implies B
T | F | F | T | T
F | T | F | T | T
T | F | F | F | T
F | T | F | F | T

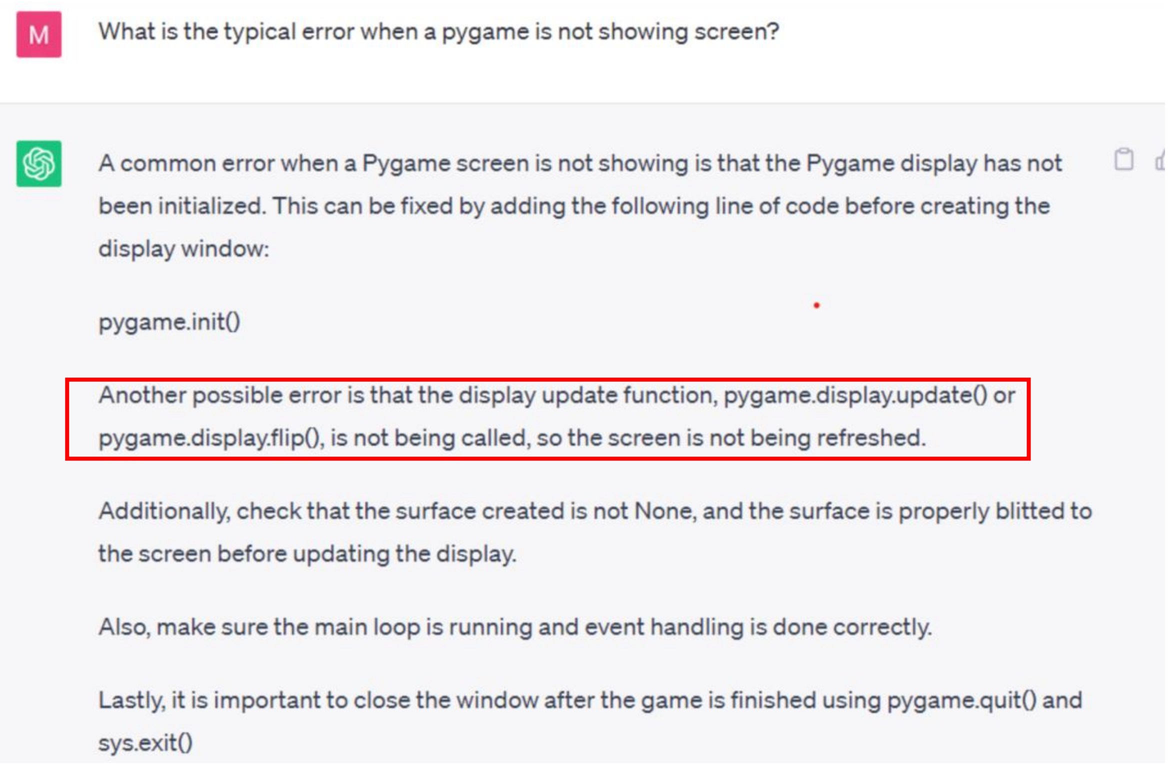
In this truth table, "falsity" is defined as "A and Not A", where A is an arbitrary proposition, and "falsity implies B" means that if the falsity condition is true (i.e. A and Not A), then B must also be true.

As you can see, the statement "falsity implies B" has value T in all rows, meaning that it is always true and can be considered as a valid principle in propositional logic.

So "ex falso quod libet" is a valid principle in propositional logic?

No, "ex falso quod libet" is not a valid principle in classical propositional logic.

Coding with ChatGPT



What is the typical error when a pygame is not showing screen?

A common error when a Pygame screen is not showing is that the Pygame display has not been initialized. This can be fixed by adding the following line of code before creating the display window:

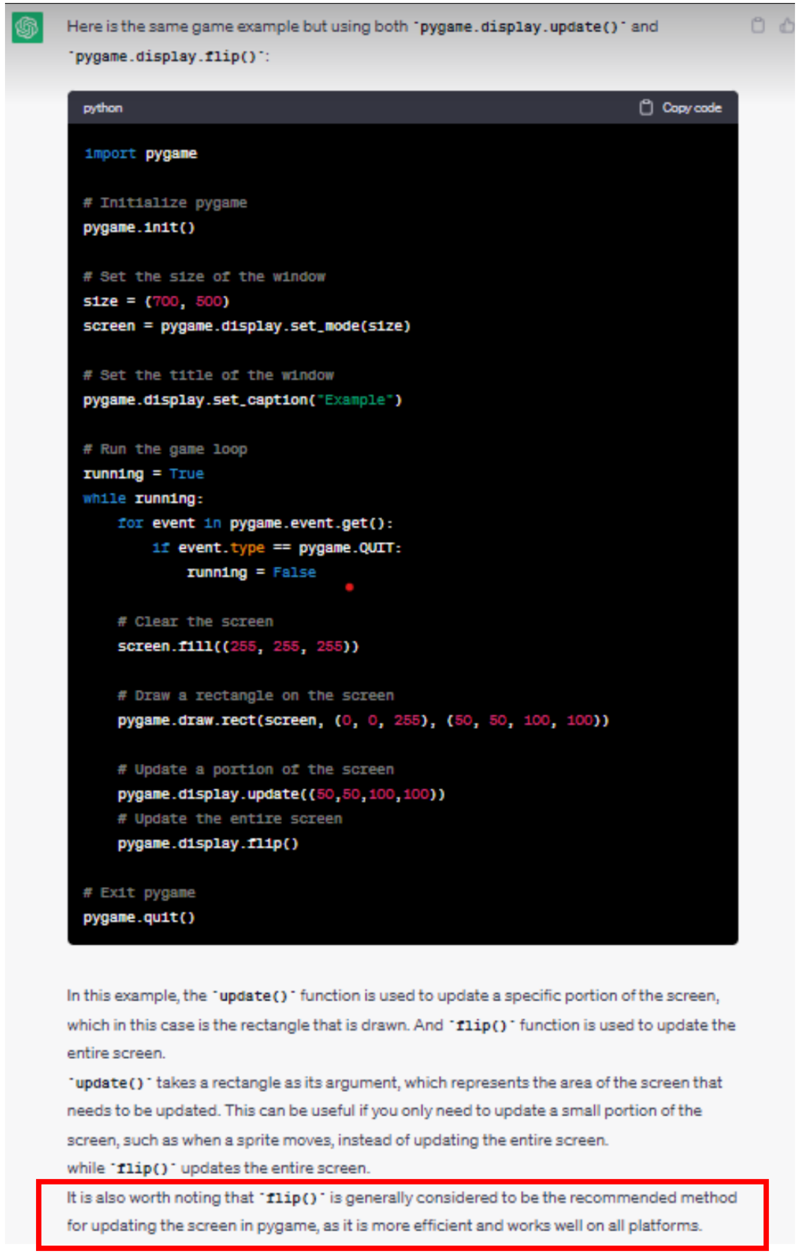
pygame.init()

Another possible error is that the display update function, pygame.display.update() or pygame.display.flip(), is not being called, so the screen is not being refreshed.

Additionally, check that the surface created is not None, and the surface is properly blitted to the screen before updating the display.

Also, make sure the main loop is running and event handling is done correctly.

Lastly, it is important to close the window after the game is finished using pygame.quit() and sys.exit()



Here is the same game example but using both `pygame.display.update()` and `pygame.display.flip()`:

```python
import pygame

# Initialize pygame
pygame.init()

# Set the size of the window
size = (700, 500)
screen = pygame.display.set_mode(size)

# Set the title of the window
pygame.display.set_caption("Example")

# Run the game loop
running = True
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

    # Clear the screen
    screen.fill((255, 255, 255))

    # Draw a rectangle on the screen
    pygame.draw.rect(screen, (0, 0, 255), (50, 50, 100, 100))

    # Update a portion of the screen
    pygame.display.update((50,50,100,100))
    # Update the entire screen
    pygame.display.flip()

# Exit pygame
pygame.quit()
```

In this example, the `update()` function is used to update a specific portion of the screen, which in this case is the rectangle that is drawn. And `flip()` function is used to update the entire screen.

`update()` takes a rectangle as its argument, which represents the area of the screen that needs to be updated. This can be useful if you only need to update a small portion of the screen, such as when a sprite moves, instead of updating the entire screen.

while `flip()` updates the entire screen.

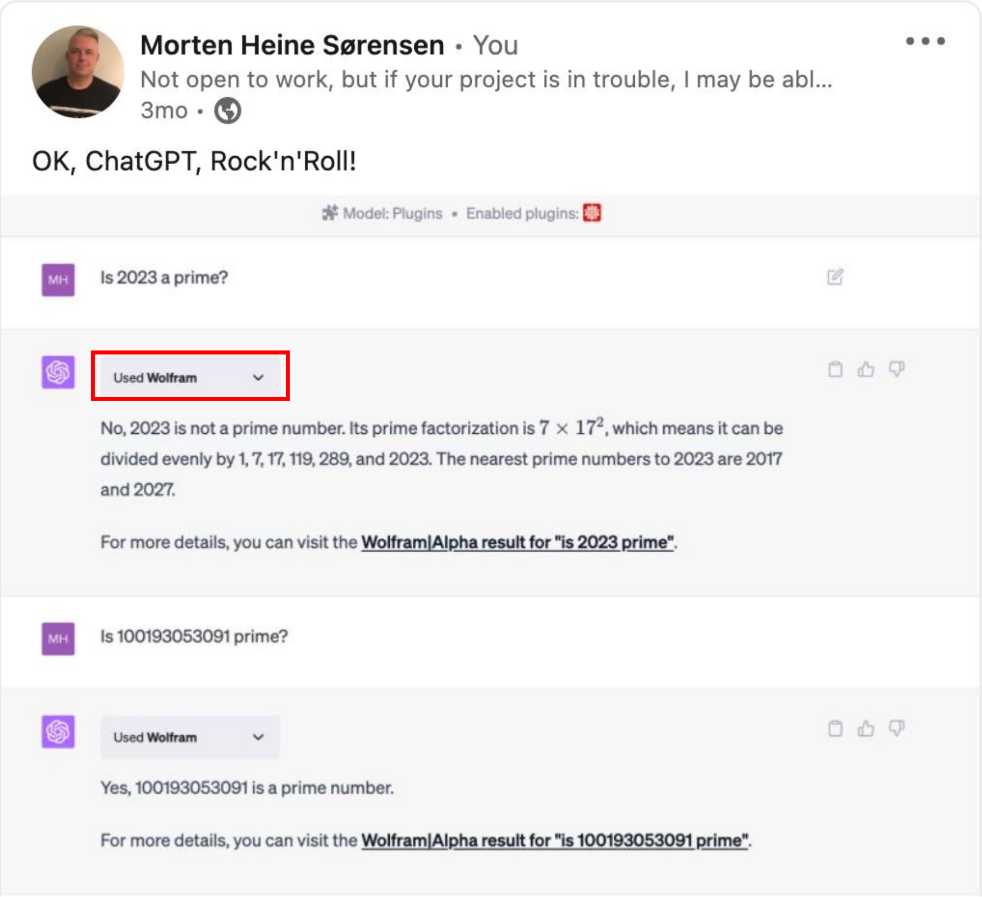
It is also worth noting that `flip()` is generally considered to be the recommended method for updating the screen in pygame, as it is more efficient and works well on all platforms.

Conclusions, February 2023

- ChatGPT does not do calculations, ChatGPT tells STORIES
- ChatGPT does not execute code , ChatGPT tells STORIES
- ChatGPT does not do symbolic reasoning , ChatGPT tells STORIES
- ChatGPT makes factual errors
- ChatGPT gives inconsistent answers
- ChatGPT tells STORIES
- But: ChatGPT can help fixing bugs **faster than Google**

Phase 2 – Late March 2023
ChatGPT4
Transition

Using plugins



Morten Heine Sørensen • You

Not open to work, but if your project is in trouble, I may be abl...

3mo •

OK, ChatGPT, Rock'n'Roll!

Is 2023 a prime?

No, 2023 is not a prime number. Its prime factorization is 7×17^2, which means it can be divided evenly by 1, 7, 17, 119, 289, and 2023. The nearest prime numbers to 2023 are 2017 and 2027.

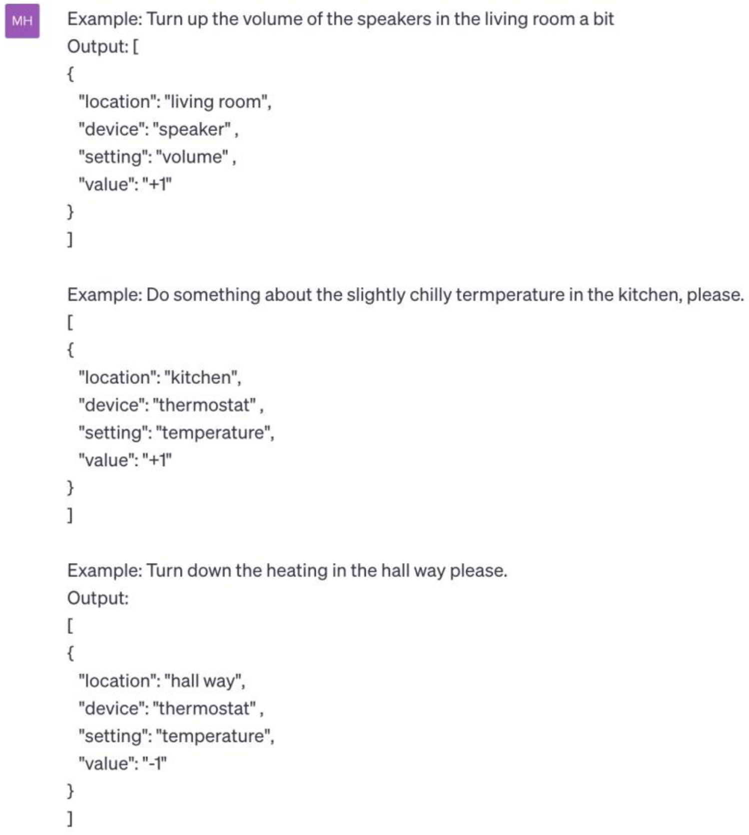
For more details, you can visit the **Wolfram|Alpha result for "is 2023 prime"**.

Is 100193053091 prime?

Yes, 100193053091 is a prime number.

For more details, you can visit the **Wolfram|Alpha result for "is 100193053091 prime"**.

Open the pod bay doors, HAL



Example: Turn up the volume of the speakers in the living room a bit

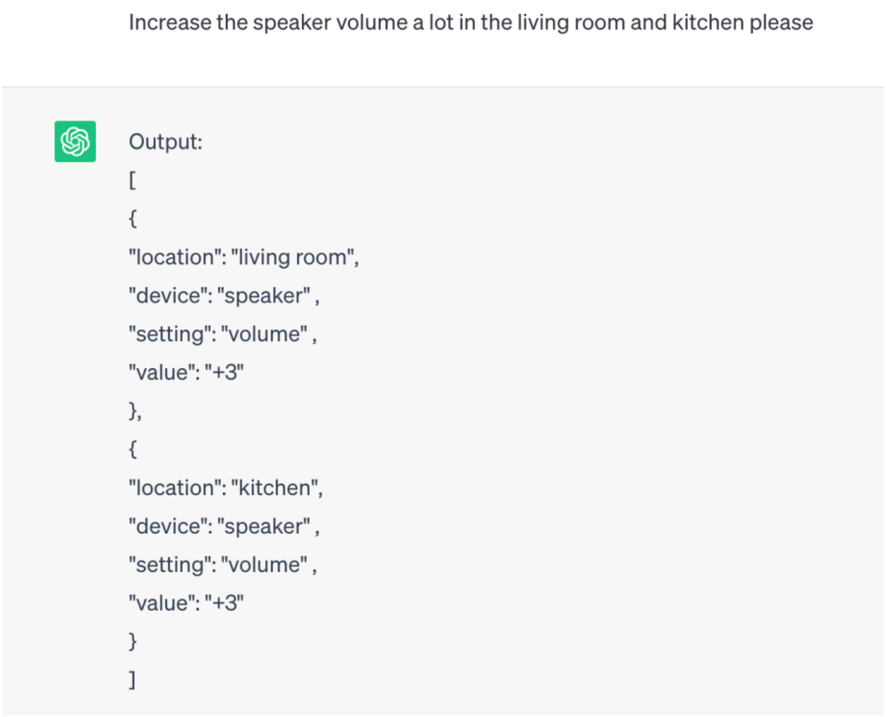
```
Output: [
{
  "location": "living room",
  "device": "speaker" ,
  "setting": "volume" ,
  "value": "+1"
}
]
```

Example: Do something about the slightly chilly termperature in the kitchen, please.

```
[
{
  "location": "kitchen",
  "device": "thermostat" ,
  "setting": "temperature",
  "value": "+1"
}
]
```

Example: Turn down the heating in the hall way please.

```
Output:
[
{
  "location": "hall way",
  "device": "thermostat" ,
  "setting": "temperature",
  "value": "-1"
}
]
```

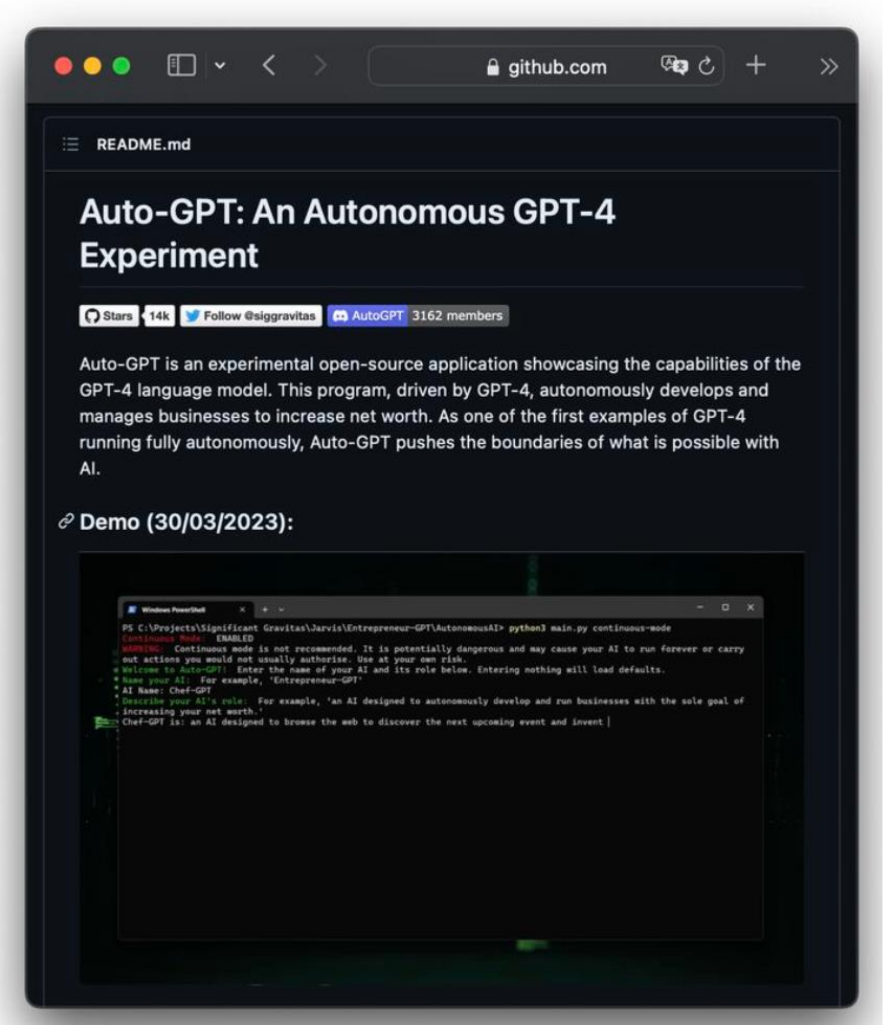


Increase the speaker volume a lot in the living room and kitchen please

```
Output:
[
{
"location": "living room",
"device": "speaker" ,
"setting": "volume" ,
"value": "+3"
},
{
"location": "kitchen",
"device": "speaker" ,
"setting": "volume" ,
"value": "+3"
}
]
```

Open AI API

- Ability to call ChatGPT
- An ocean of possibilities for apps



Auto-GPT: An Autonomous GPT-4 Experiment

Auto-GPT is an experimental open-source application showcasing the capabilities of the GPT-4 language model. This program, driven by GPT-4, autonomously develops and manages businesses to increase net worth. As one of the first examples of GPT-4 running fully autonomously, Auto-GPT pushes the boundaries of what is possible with AI.

Demo (30/03/2023):

```
PS C:\Projects\Significant Gravitas\Jarvis\Entrepreneur-GPT\AutonomousAI> python3 main.py continuous-mode
Continuous Mode:  ENABLED
WARNING:  Continuous mode is not recommended. It is potentially dangerous and may cause your AI to run forever or carry
out actions you would not usually authorise. Use at your own risk.
Welcome to Auto-GPT!  Enter the name of your AI and its role below. Entering nothing will load defaults.
Name your AI:  For example, 'Entrepreneur-GPT'
AI Name: Chef-GPT
Describe your AI's role:  For example, 'an AI designed to autonomously develop and run businesses with the sole goal of
increasing your net worth.'
Chef-GPT is: an AI designed to browse the web to discover the next upcoming event and invent
```

Conclusions, May 2023

- ChatGPT delegates to plugins
    - They can do calculations
    - They can execute code
    - They can do symbolic reasoning
- You can call its API and use the response to call other APIs
- Overall model provides better answers
- ChatGPT still tells stories – *this is the model*
- The model is an *advantage*

Phase 3 – Summer 2023
ChatGPT4
New Era

LinkedIn rumours

- Experienced developer 1:
  - ChatGPT is a factor *x* quicker coder than me.
- Experienced developer 2:
  - Last year if I got $*x* mio for startup, I’d hire a team of devs.
  - Now I would use ChatGPT.
- Prompt Engineering
  - Lots of fuzz.

Reported experiments (selection)

- **Danusha Navod Gamage: 7 Ways ChatGPT Can Help Developers.**
  - Generate code snippets
  - Write test cases
  - Fix defects
  - Etc.
- **Martin Fowler: An example of LLM prompting for programming.**
  - Add a feature to an enterprise application
- **Mike Loukides: Real World Programming with ChatGPT.**
  - Same as above.
- **David Gewirtz: How to use ChatGPT to write code.**
  - Partial implementation of browser plugin to play tunes matching research articles.
- **Philip Guo: Real-Real-World Programming with ChatGPT.**
  - Wordpress plugin.

Scaling Real-world application development

LESSON 1: ChatGPT can provide tutorials on how to accomplish tasks that are tailored to your specific application, making them very easy and fast to use.

Full-stack representative application:

- UI built with React.
- API built on Node.js.
- Data layer with Postgres database.

Approach:

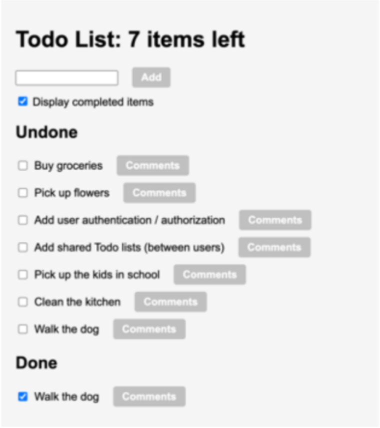
LESSON 3: Split the development into sprints and user stories, like you would with a team of developers. Then proceed sprint by sprint, user story by user story, for the same reasons as you normally do.

- Standard approach with developers
  - Sprints
  - Releases
  - Environments
  - Pull-requests
  - User-stories
  - UI Specs
  - DB and API specs
- **Delegate user stories/specs to ChatGPT**
- **Receive pull-requests from ChatGPT**

Steps:

1. Set up the project
2. Create the data model
3. Create the API
4. Create the UI
5. Create the styling
6. Create the tests
7. Refactor common parts
8. Mature the application
9. Set up environments+pipelines
10. Scale with formalism
11. Scale with developers

LESSON 5: Except for mock-ups, UI experimentation and hobby projects, most sprints should start by implementing the data model of the feature.



Todo List: 7 items left

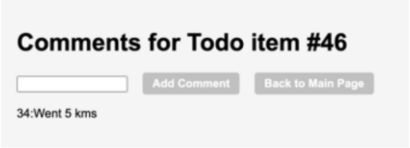
☑ Display completed items

Undone

☐ Buy groceries
☐ Pick up flowers
☐ Add user authentication / authorization
☐ Add shared Todo lists (between users)
☐ Pick up the kids in school
☐ Clean the kitchen
☐ Walk the dog

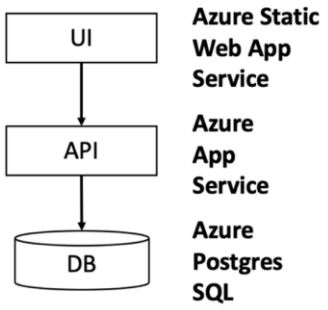
Done

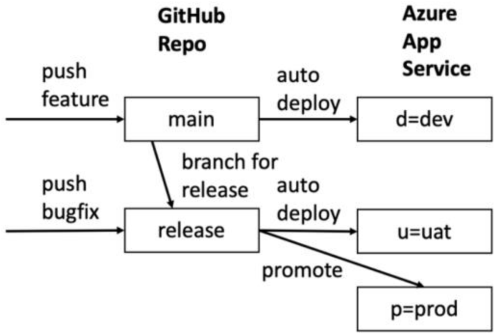
☑ Walk the dog



Comments for Todo item #46

34:Went 5 kms





UI	API
React React Route	JS Express SQL
UI Test Puppeteer	API Test Axios
Node.js	Node.js

Dialogues – Set up project

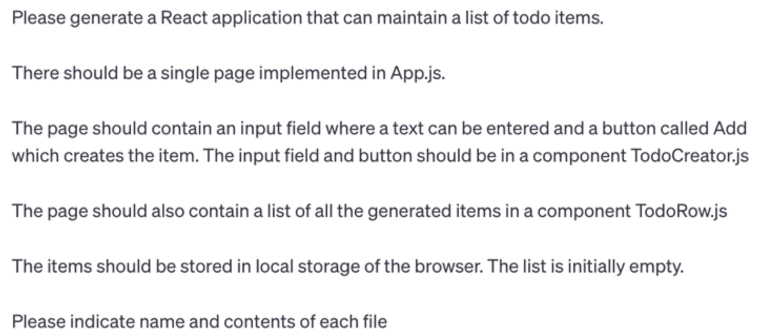
- Install node
- Install IDE
- npx create-react-app
  - Project
  - File structure
  - Skeleton
- Run the sample application

LESSON 2: Spend the time needed to set up the local development environment to efficiently support the iterations with ChatGPT. It does not matter if this take a bit more time than ideal since it will be a one-time cost.

Dialogues – First part of UI

LESSON 4: In Sprint 1, establish the fundamental architecture of the application, for instance a UI layer, an API layer, and the DB layer developed well enough to cover a small feature. The fundamental architecture should not only align to functional and non-functional requirements, but also to the team size and team experience.

LESSON 8: When working with ChatGPT on User Interface, ask first for a simple version. Then add remaining controls one by one or in small groups.



Please generate a React application that can maintain a list of todo items.

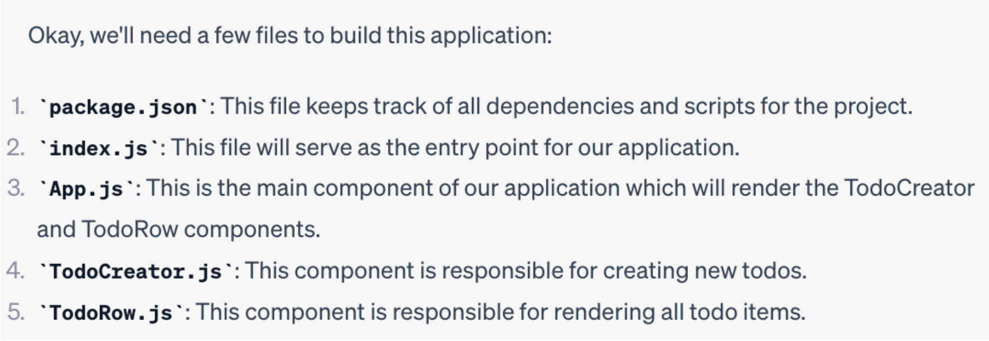
There should be a single page implemented in App.js.

The page should contain an input field where a text can be entered and a button called Add which creates the item. The input field and button should be in a component TodoCreator.js

The page should also contain a list of all the generated items in a component TodoRow.js

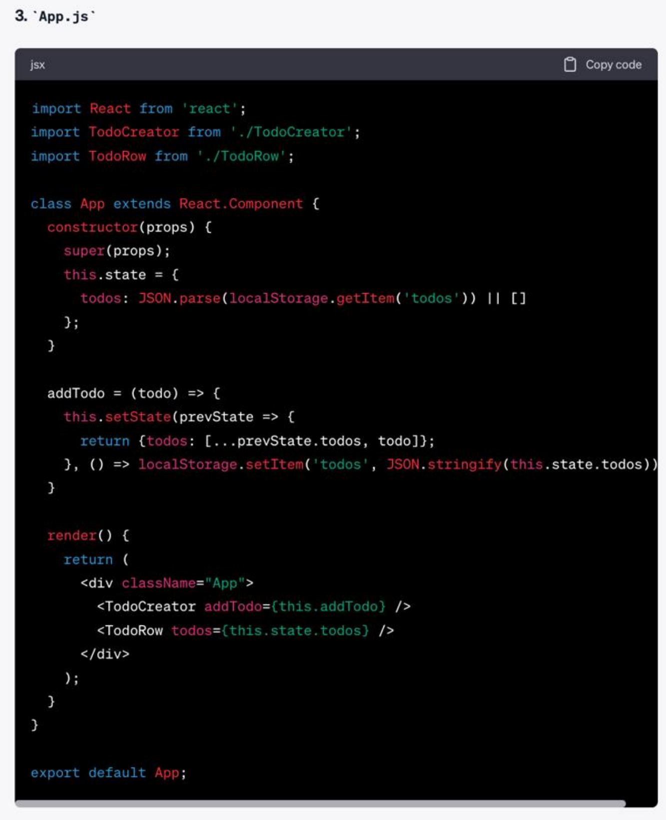
The items should be stored in local storage of the browser. The list is initially empty.

Please indicate name and contents of each file



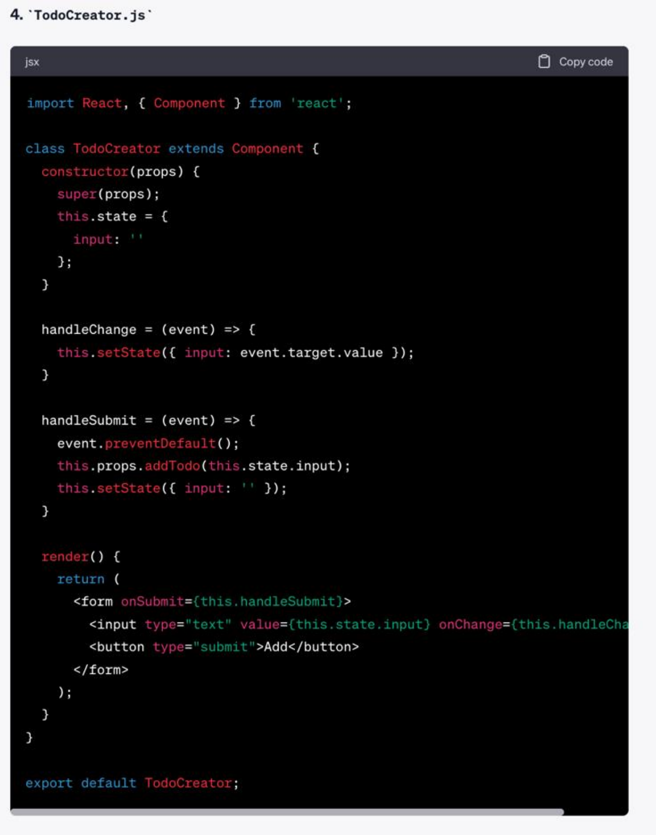
Okay, we'll need a few files to build this application:

1. `package.json`: This file keeps track of all dependencies and scripts for the project.
2. `index.js`: This file will serve as the entry point for our application.
3. `App.js`: This is the main component of our application which will render the TodoCreator and TodoRow components.
4. `TodoCreator.js`: This component is responsible for creating new todos.
5. `TodoRow.js`: This component is responsible for rendering all todo items.



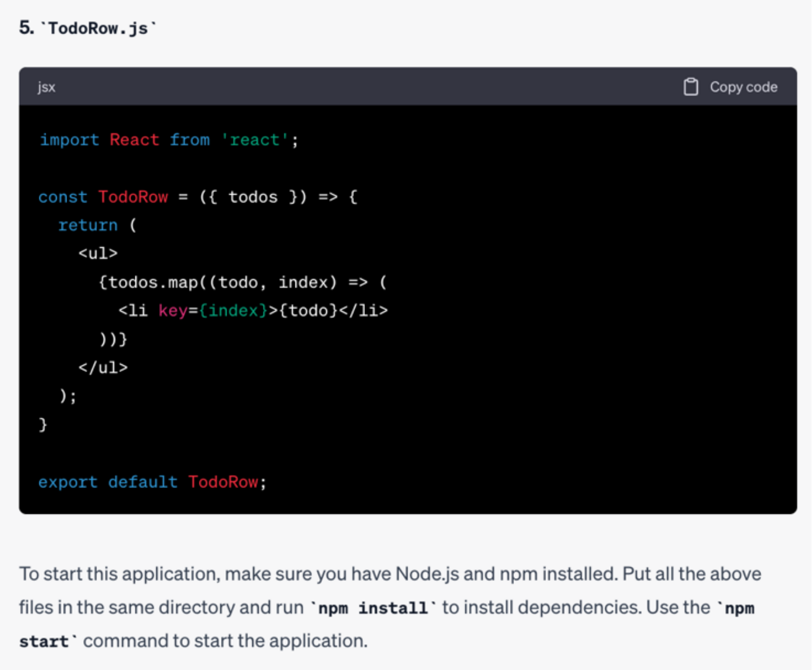
3. `App.js`

```jsx
import React from 'react';
import TodoCreator from './TodoCreator';
import TodoRow from './TodoRow';

class App extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      todos: JSON.parse(localStorage.getItem('todos')) || []
    };
  }

  addTodo = (todo) => {
    this.setState(prevState => {
      return {todos: [...prevState.todos, todo]};
    }, () => localStorage.setItem('todos', JSON.stringify(this.state.todos))
  }

  render() {
    return (
      <div className="App">
        <TodoCreator addTodo={this.addTodo} />
        <TodoRow todos={this.state.todos} />
      </div>
    );
  }
}

export default App;
```



4. `TodoCreator.js`

```jsx
import React, { Component } from 'react';

class TodoCreator extends Component {
  constructor(props) {
    super(props);
    this.state = {
      input: ''
    };
  }

  handleChange = (event) => {
    this.setState({ input: event.target.value });
  }

  handleSubmit = (event) => {
    event.preventDefault();
    this.props.addTodo(this.state.input);
    this.setState({ input: '' });
  }

  render() {
    return (
      <form onSubmit={this.handleSubmit}>
        <input type="text" value={this.state.input} onChange={this.handleCha
        <button type="submit">Add</button>
      </form>
    );
  }
}

export default TodoCreator;
```



5. `TodoRow.js`

```jsx
import React from 'react';

const TodoRow = ({ todos }) => {
  return (
    <ul>
      {todos.map((todo, index) => (
        <li key={index}>{todo}</li>
      ))}
    </ul>
  );
}

export default TodoRow;
```

To start this application, make sure you have Node.js and npm installed. Put all the above files in the same directory and run `npm install` to install dependencies. Use the `npm start` command to start the application.

LESSON 6: ChatGPT's answers should be evaluated like they come from another (sometimes less experienced) person, rather than from a flawless machine. The human developer who integrates ChatGPT's responses into the application remains liable.

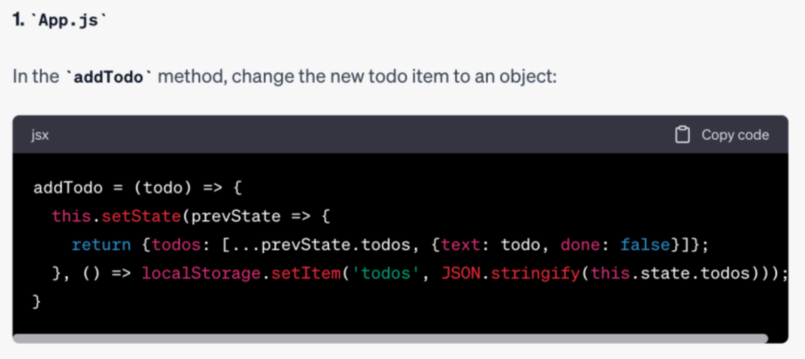
LESSON 10: When starting the interaction with ChatGPT, have a dummy application working.

Dialogues – Todo done checkmark

In the todo list, there should be a editable check mark indicating whether it is done or not.

Please indicate name and contents of files that need change.

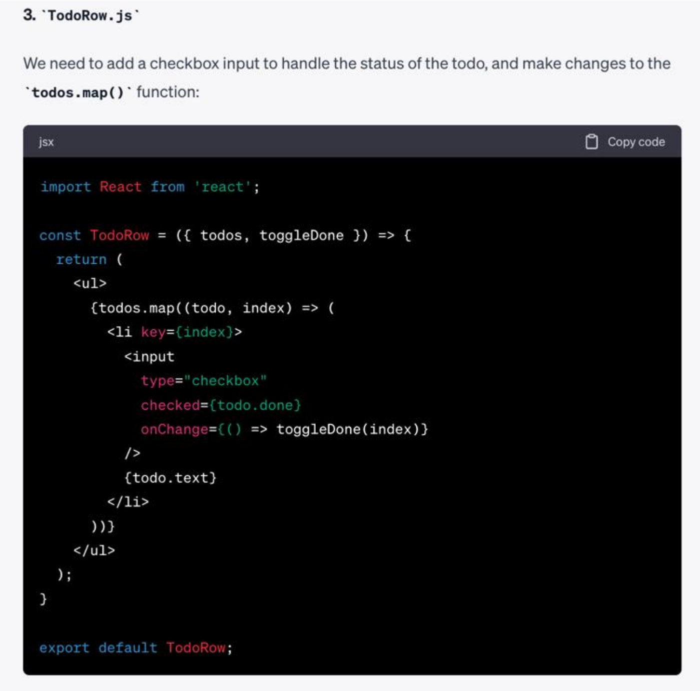
LESSON 9: Ask for full contents of files. It speeds up the process.



1. `App.js`

In the `addTodo` method, change the new todo item to an object:

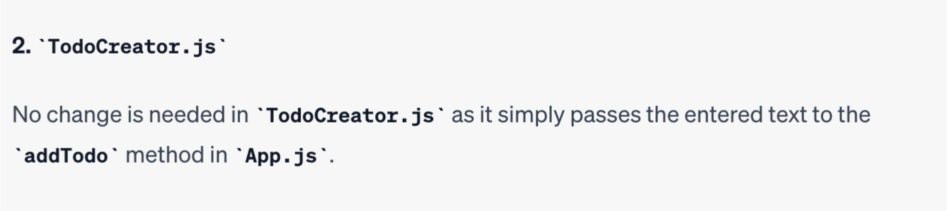
```jsx
addTodo = (todo) => {
  this.setState(prevState => {
    return {todos: [...prevState.todos, {text: todo, done: false}]};
  }, () => localStorage.setItem('todos', JSON.stringify(this.state.todos)));
}
```



3. `TodoRow.js`

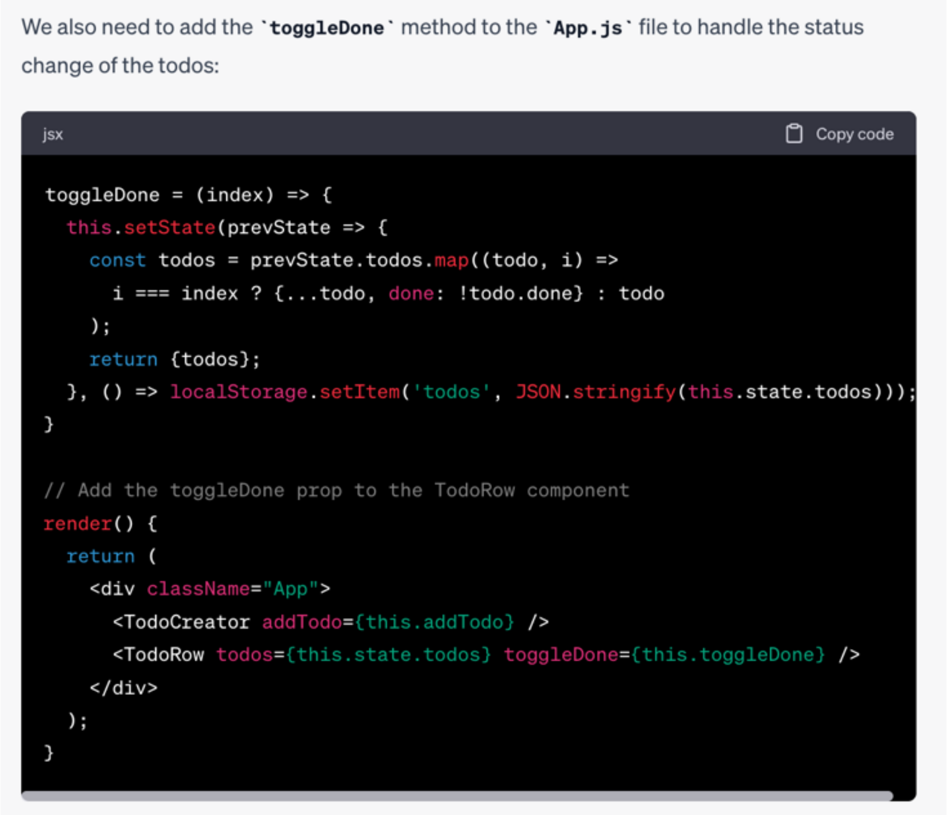
We need to add a checkbox input to handle the status of the todo, and make changes to the `todos.map()` function:

```jsx
import React from 'react';

const TodoRow = ({ todos, toggleDone }) => {
  return (
    <ul>
      {todos.map((todo, index) => (
        <li key={index}>
          <input
            type="checkbox"
            checked={todo.done}
            onChange={() => toggleDone(index)}
          />
          {todo.text}
        </li>
      ))}
    </ul>
  );
}

export default TodoRow;
```



2. `TodoCreator.js`

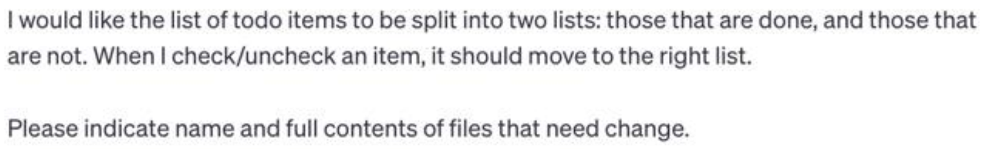
No change is needed in `TodoCreator.js` as it simply passes the entered text to the `addTodo` method in `App.js`.



We also need to add the `toggleDone` method to the `App.js` file to handle the status change of the todos:

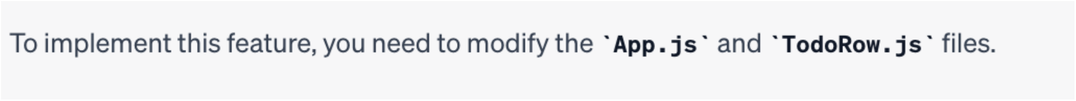
```jsx
toggleDone = (index) => {
  this.setState(prevState => {
    const todos = prevState.todos.map((todo, i) =>
      i === index ? {...todo, done: !todo.done} : todo
    );
    return {todos};
  }, () => localStorage.setItem('todos', JSON.stringify(this.state.todos)));
}

// Add the toggleDone prop to the TodoRow component
render() {
  return (
    <div className="App">
      <TodoCreator addTodo={this.addTodo} />
      <TodoRow todos={this.state.todos} toggleDone={this.toggleDone} />
    </div>
  );
}
```

Dialogues – Separate done and undone items



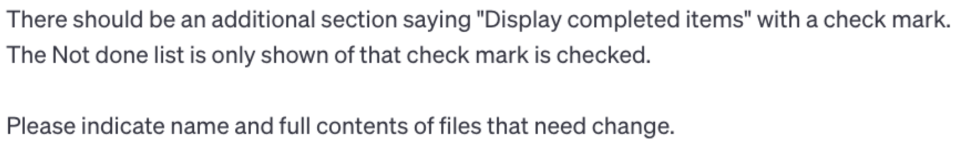
I would like the list of todo items to be split into two lists: those that are done, and those that are not. When I check/uncheck an item, it should move to the right list.

Please indicate name and full contents of files that need change.



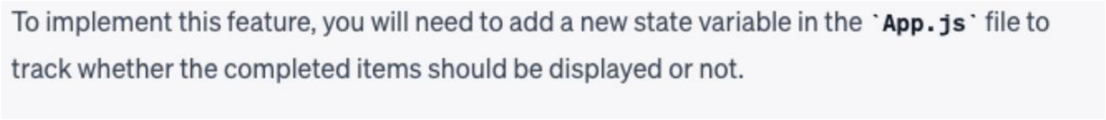
To implement this feature, you need to modify the `App.js` and `TodoRow.js` files.

Details omitted



There should be an additional section saying "Display completed items" with a check mark. The Not done list is only shown of that check mark is checked.

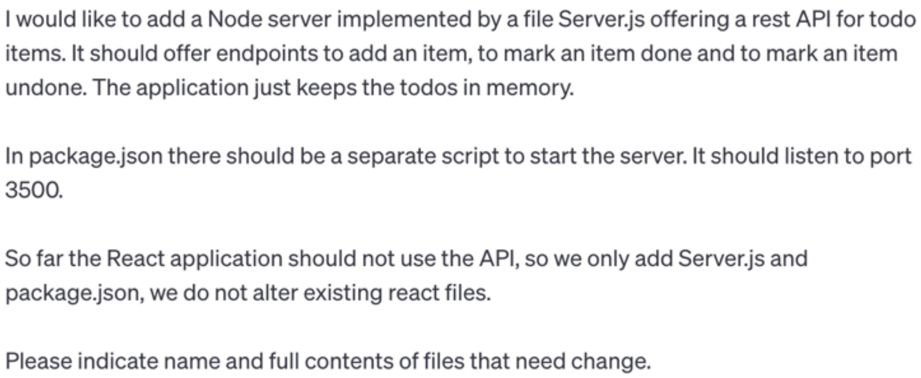
Please indicate name and full contents of files that need change.



To implement this feature, you will need to add a new state variable in the `App.js` file to track whether the completed items should be displayed or not.

Details omitted

Dialogues – Create the server



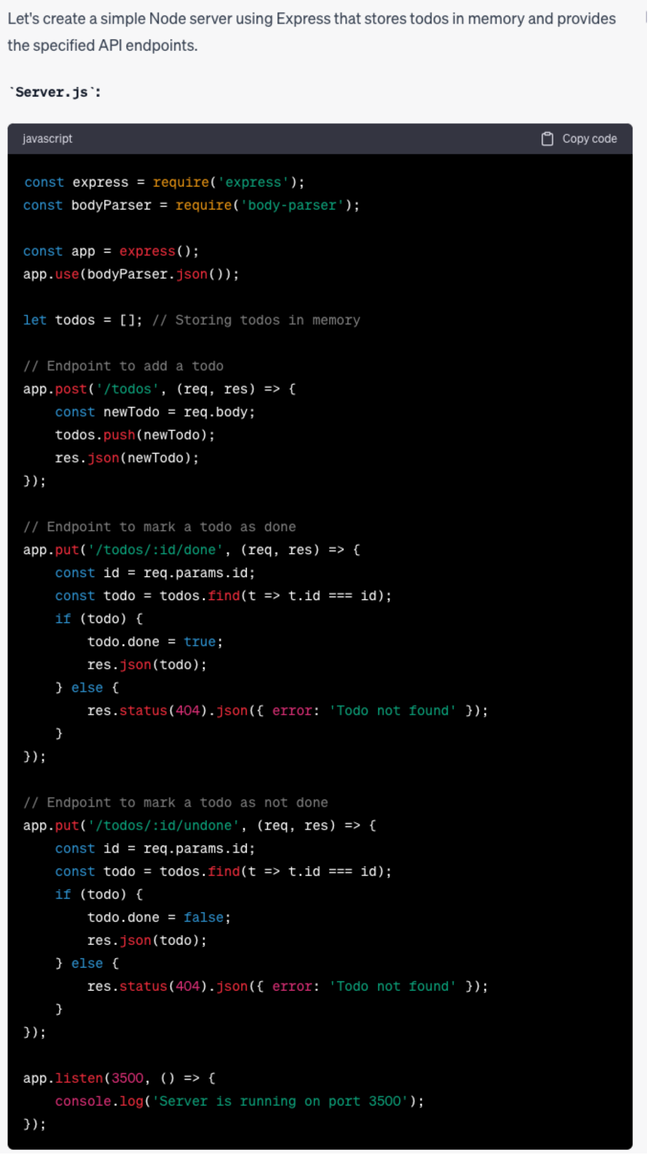
I would like to add a Node server implemented by a file Server.js offering a rest API for todo items. It should offer endpoints to add an item, to mark an item done and to mark an item undone. The application just keeps the todos in memory.

In package.json there should be a separate script to start the server. It should listen to port 3500.

So far the React application should not use the API, so we only add Server.js and package.json, we do not alter existing react files.

Please indicate name and full contents of files that need change.

LESSON 7: When starting the API, get a simple server running with a simple example and make sure you are happy with it to some level of maturity. There may be some production hardening missing that can be covered latter, but the basic structure should be correct and satisfactory.

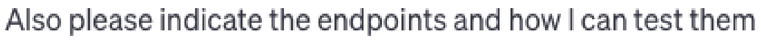


Let's create a simple Node server using Express that stores todos in memory and provides the specified API endpoints.

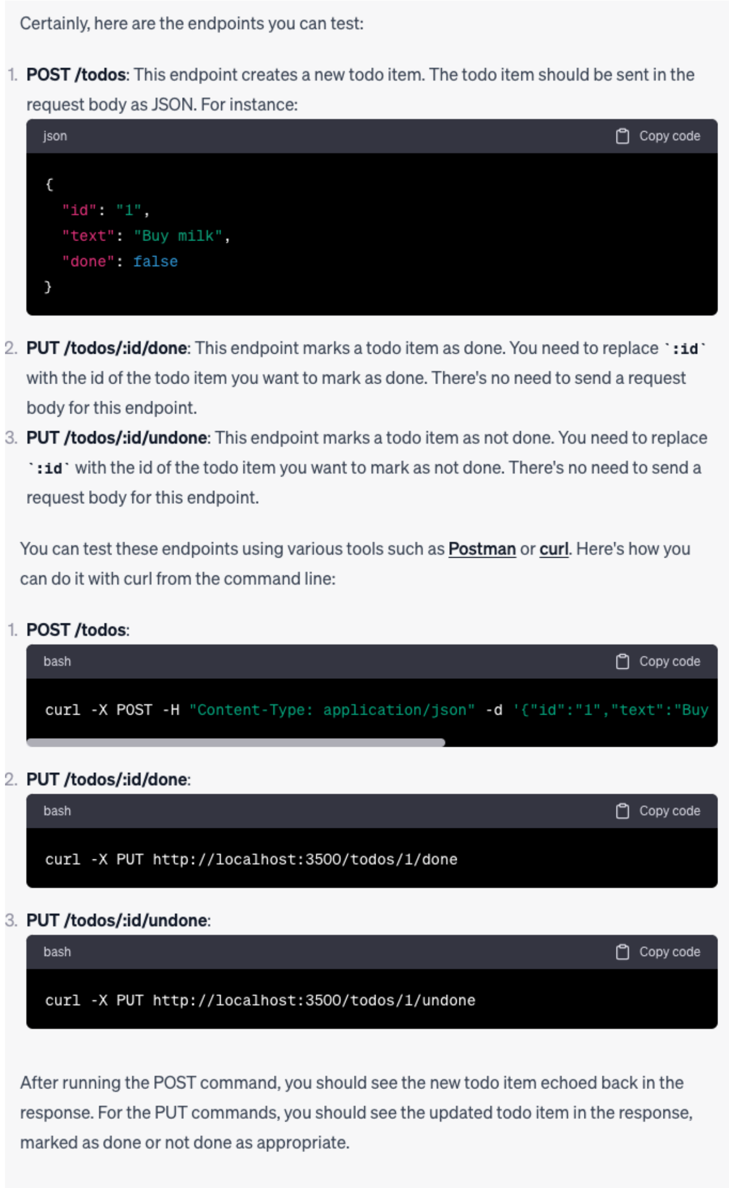
`Server.js`:

```javascript
const express = require('express');
const bodyParser = require('body-parser');

const app = express();
app.use(bodyParser.json());

let todos = []; // Storing todos in memory

// Endpoint to add a todo
app.post('/todos', (req, res) => {
    const newTodo = req.body;
    todos.push(newTodo);
    res.json(newTodo);
});

// Endpoint to mark a todo as done
app.put('/todos/:id/done', (req, res) => {
    const id = req.params.id;
    const todo = todos.find(t => t.id === id);
    if (todo) {
        todo.done = true;
        res.json(todo);
    } else {
        res.status(404).json({ error: 'Todo not found' });
    }
});

// Endpoint to mark a todo as not done
app.put('/todos/:id/undone', (req, res) => {
    const id = req.params.id;
    const todo = todos.find(t => t.id === id);
    if (todo) {
        todo.done = false;
        res.json(todo);
    } else {
        res.status(404).json({ error: 'Todo not found' });
    }
});

app.listen(3500, () => {
    console.log('Server is running on port 3500');
});
```

Dialogues – Testing the server



Also please indicate the endpoints and how I can test them



Certainly, here are the endpoints you can test:

1. **POST /todos**: This endpoint creates a new todo item. The todo item should be sent in the request body as JSON. For instance:

```json
{
  "id": "1",
  "text": "Buy milk",
  "done": false
}
```

2. **PUT /todos/:id/done**: This endpoint marks a todo item as done. You need to replace `:id` with the id of the todo item you want to mark as done. There's no need to send a request body for this endpoint.
3. **PUT /todos/:id/undone**: This endpoint marks a todo item as not done. You need to replace `:id` with the id of the todo item you want to mark as not done. There's no need to send a request body for this endpoint.

You can test these endpoints using various tools such as Postman or curl. Here's how you can do it with curl from the command line:

1. **POST /todos**:

```bash
curl -X POST -H "Content-Type: application/json" -d '{"id":"1","text":"Buy
```

2. **PUT /todos/:id/done**:

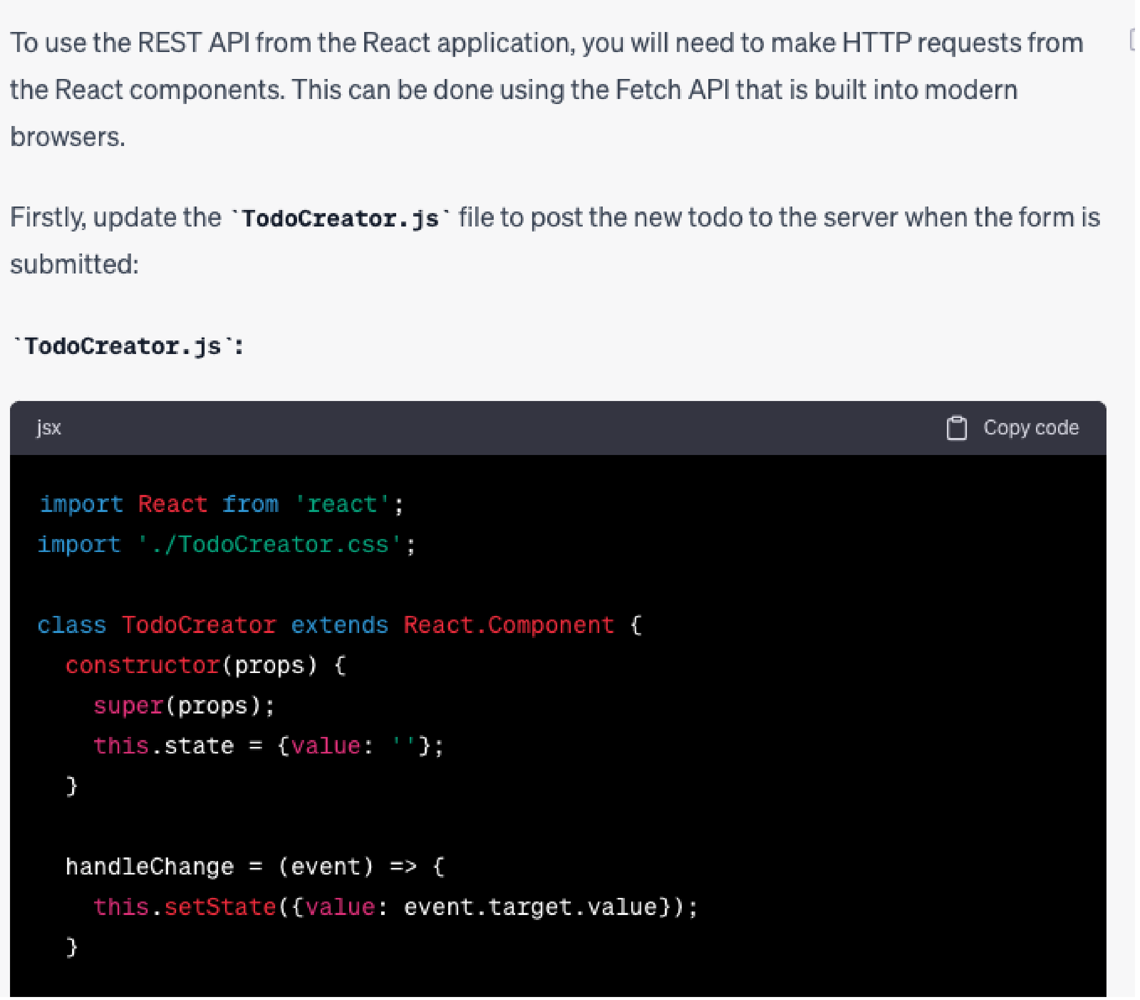
```bash
curl -X PUT http://localhost:3500/todos/1/done
```

3. **PUT /todos/:id/undone**:

```bash
curl -X PUT http://localhost:3500/todos/1/undone
```

After running the POST command, you should see the new todo item echoed back in the response. For the PUT commands, you should see the updated todo item in the response, marked as done or not done as appropriate.

Dialogues – Using the endpoints from the UI



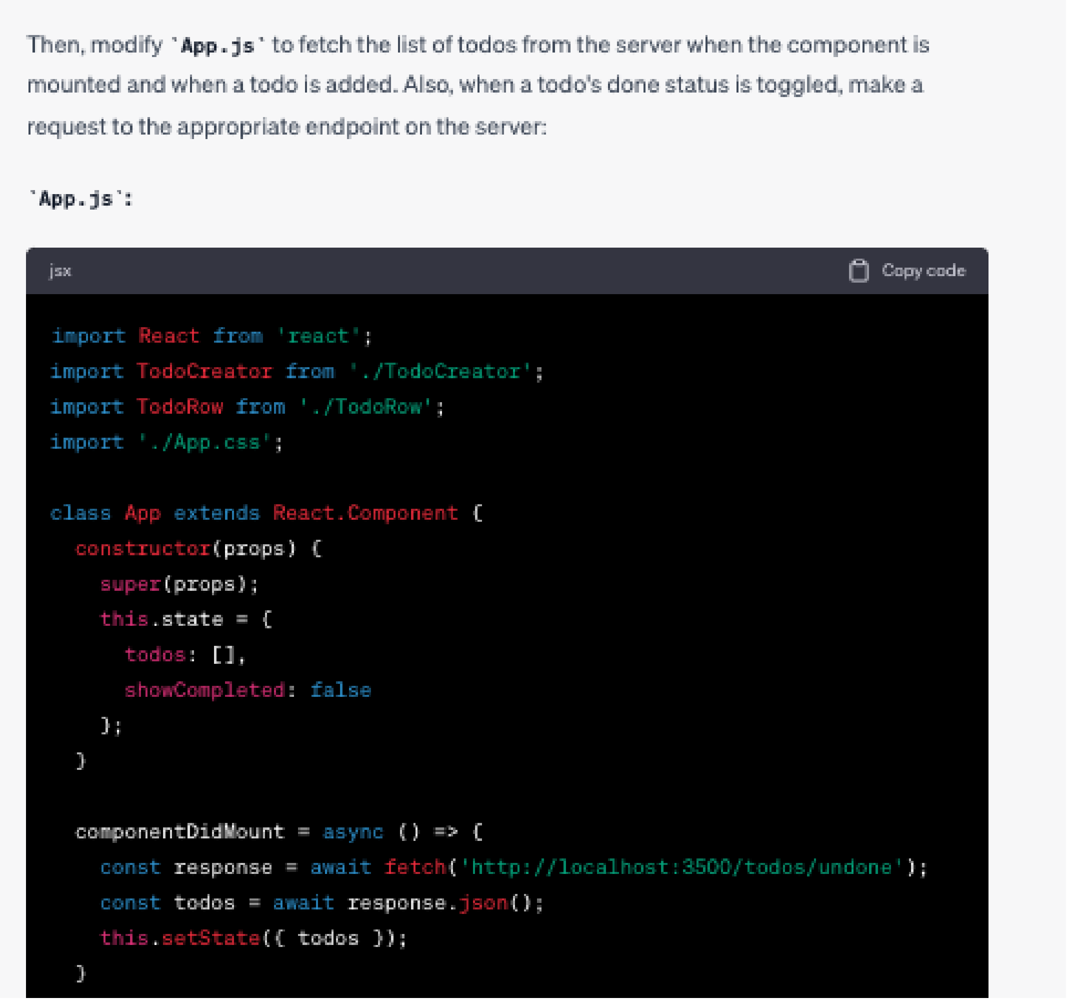
To use the REST API from the React application, you will need to make HTTP requests from the React components. This can be done using the Fetch API that is built into modern browsers.

Firstly, update the `TodoCreator.js` file to post the new todo to the server when the form is submitted:

`TodoCreator.js`:

```jsx
import React from 'react';
import './TodoCreator.css';

class TodoCreator extends React.Component {
  constructor(props) {
    super(props);
    this.state = {value: ''};
  }

  handleChange = (event) => {
    this.setState({value: event.target.value});
  }
```

Details omitted



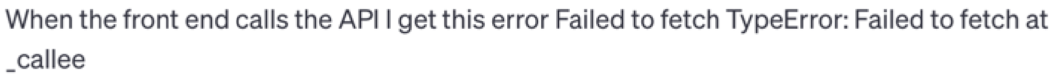
Then, modify `App.js` to fetch the list of todos from the server when the component is mounted and when a todo is added. Also, when a todo's done status is toggled, make a request to the appropriate endpoint on the server:

`App.js`:

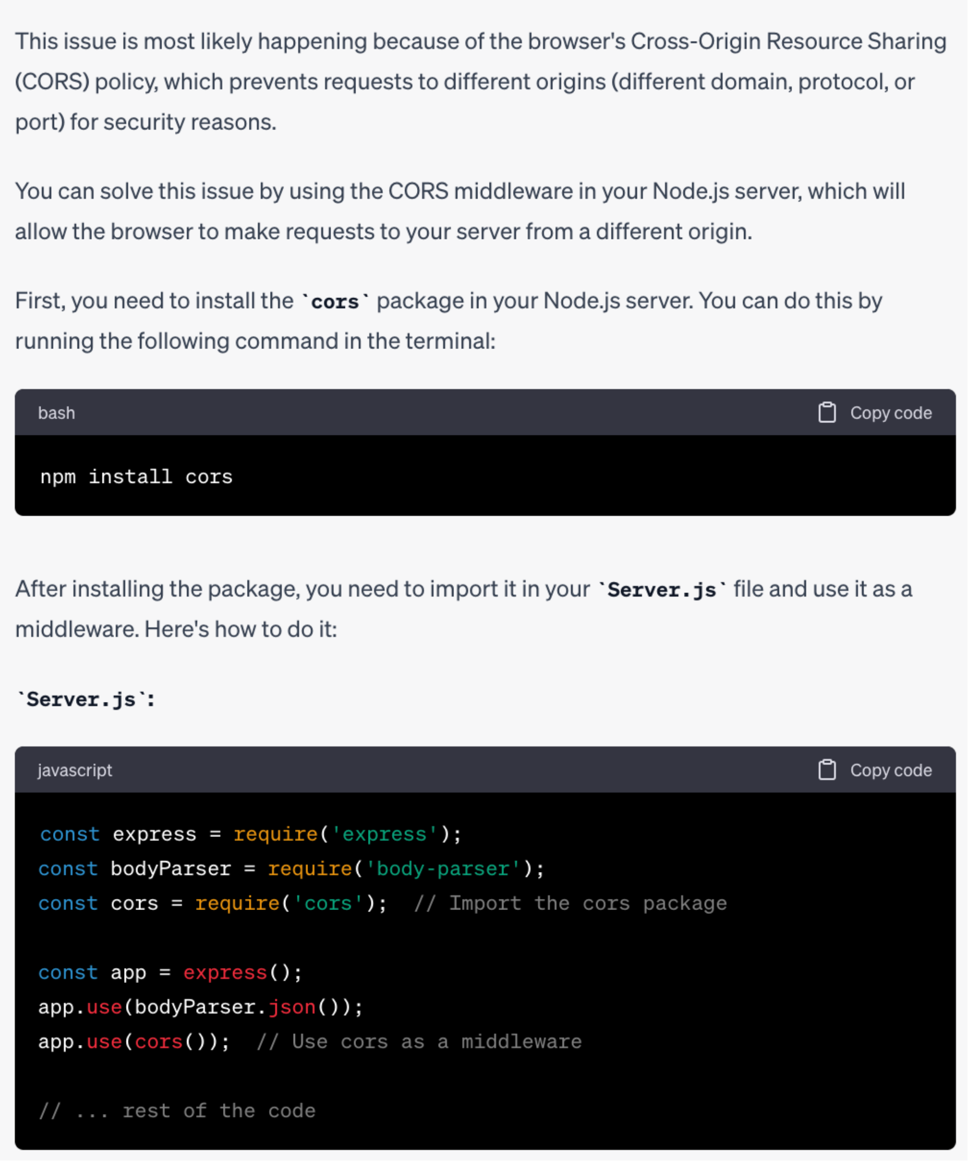
```jsx
import React from 'react';
import TodoCreator from './TodoCreator';
import TodoRow from './TodoRow';
import './App.css';

class App extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      todos: [],
      showCompleted: false
    };
  }

  componentDidMount = async () => {
    const response = await fetch('http://localhost:3500/todos/undone');
    const todos = await response.json();
    this.setState({ todos });
  }
```

Details omitted

Dialogues – An error



When the front end calls the API I get this error Failed to fetch TypeError: Failed to fetch at _callee



This issue is most likely happening because of the browser's Cross-Origin Resource Sharing (CORS) policy, which prevents requests to different origins (different domain, protocol, or port) for security reasons.

You can solve this issue by using the CORS middleware in your Node.js server, which will allow the browser to make requests to your server from a different origin.

First, you need to install the `cors` package in your Node.js server. You can do this by running the following command in the terminal:

```bash
npm install cors
```

After installing the package, you need to import it in your `Server.js` file and use it as a middleware. Here's how to do it:

`Server.js`:

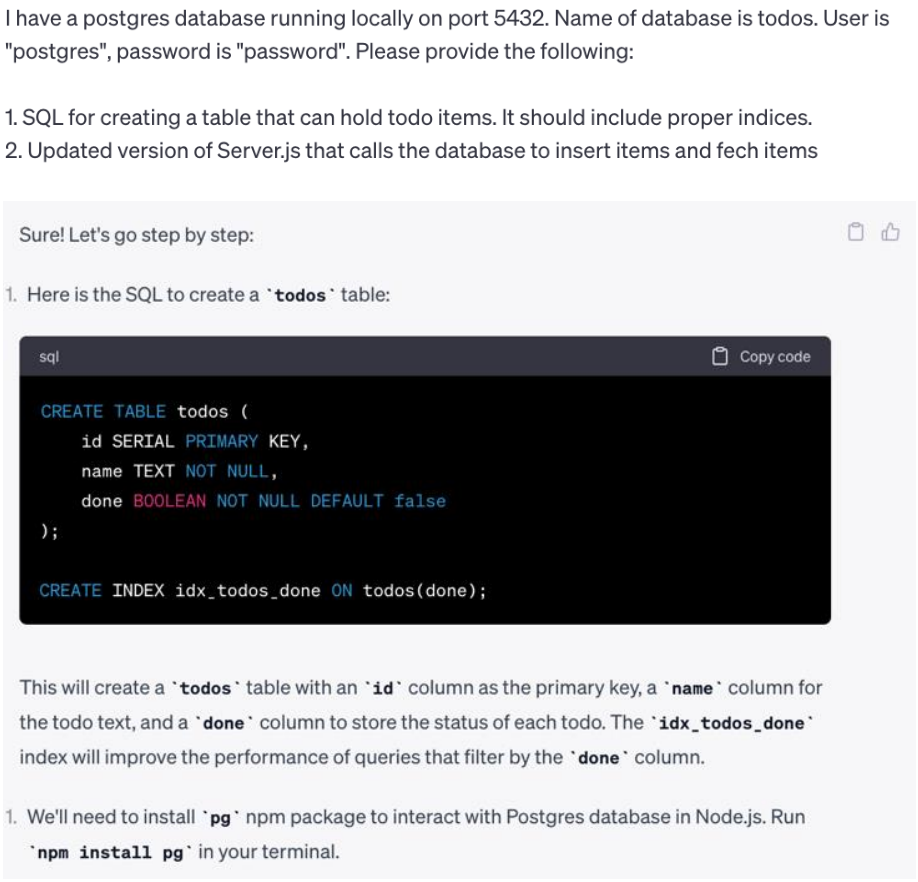
```javascript
const express = require('express');
const bodyParser = require('body-parser');
const cors = require('cors');  // Import the cors package

const app = express();
app.use(bodyParser.json());
app.use(cors());  // Use cors as a middleware

// ... rest of the code
```

LESSON 11: ChatGPT may generate code with issues for various reasons, e.g., because we did not ask explicitly to avoid them or because he made an interpretation that is incorrect in our context. In such cases, simply ask for help; most of the time he will be able to (help) resolve the problem.

LESSON 12: As mentioned, ChatGPT may generate code with errors but getting it to fix them is quicker than writing the code yourself, which may also introduce errors that may take time to find and resolve.

LESSON 13: ChatGPT may generate code with problems that were solved earlier in the dialog. You need to test what he returns at every step and ask errors to be fixed.

Dialogues – Creating the database



I have a postgres database running locally on port 5432. Name of database is todos. User is "postgres", password is "password". Please provide the following:

1. SQL for creating a table that can hold todo items. It should include proper indices.
2. Updated version of Server.js that calls the database to insert items and fech items

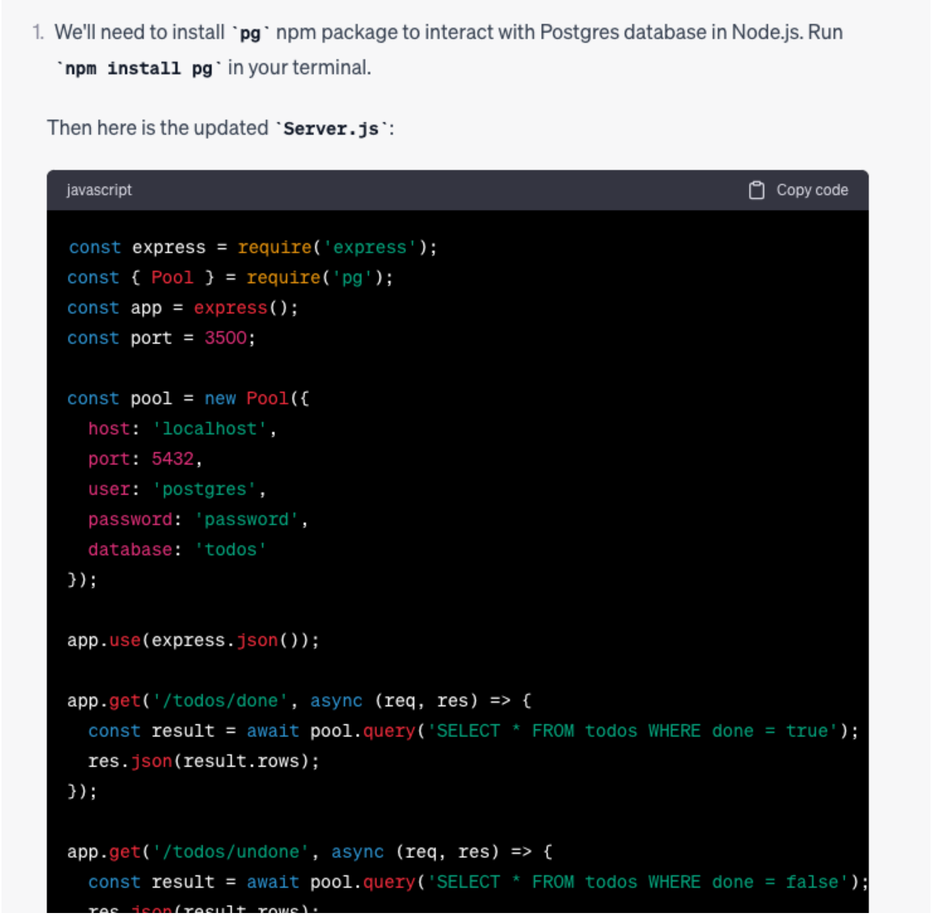
Sure! Let's go step by step:

1. Here is the SQL to create a `todos` table:

```sql
CREATE TABLE todos (
    id SERIAL PRIMARY KEY,
    name TEXT NOT NULL,
    done BOOLEAN NOT NULL DEFAULT false
);

CREATE INDEX idx_todos_done ON todos(done);
```

This will create a `todos` table with an `id` column as the primary key, a `name` column for the todo text, and a `done` column to store the status of each todo. The `idx_todos_done` index will improve the performance of queries that filter by the `done` column.

1. We'll need to install `pg` npm package to interact with Postgres database in Node.js. Run `npm install pg` in your terminal.



1. We'll need to install `pg` npm package to interact with Postgres database in Node.js. Run `npm install pg` in your terminal.

Then here is the updated `Server.js`:

```javascript
const express = require('express');
const { Pool } = require('pg');
const app = express();
const port = 3500;

const pool = new Pool({
  host: 'localhost',
  port: 5432,
  user: 'postgres',
  password: 'password',
  database: 'todos'
});

app.use(express.json());

app.get('/todos/done', async (req, res) => {
  const result = await pool.query('SELECT * FROM todos WHERE done = true');
  res.json(result.rows);
});

app.get('/todos/undone', async (req, res) => {
  const result = await pool.query('SELECT * FROM todos WHERE done = false');
  res.json(result.rows);
```

LESSON 14: Be as precise as you can in stating what you need from ChatGPT. Whenever you omit details, ChatGPT may do something else than you expect.

LESSON 15: If ChatGPT is lacking context, he may make assumptions instead of asking for clarifications. You can circumvent this by explicitly directing him initially to ask clarifications.

Dialogues – Other parts

- Create the styling
- Create the tests
- Refactor out common parts
- Mature the application
- Set up the prod environment
- Scale with formalism
- Scale with developers
- Dealing with limitations

LESSON 16: Sometimes, several iterations do not bring you closer to a solution. In these cases, consider reverting to classical techniques, like Googling, Stack Overflow, YouTube demos, reading the documentation, etc.

LESSON 17: ChatGPT currently does not know about knowledge published after September 2021

LESSON 18: ChatGPT generates several errors. It is valuable to have a slim UI test and API test that can be run locally before merging pull requests.

LESSON 20: As you review the PRs from ChatGPT, keep an eye on parts of the code that could or should be refactored. Either do the refactoring yourself or ask ChatGPT to do it.

LESSON 23: See the code by ChatGPT as your code. You must be able to account for it. That way it has same value as code developed by you or your developers.

LESSON 24: The code that you get from ChatGPT may not be ready for production, but he can help you understand what needs to be checked and how, and which changes are needed as a result of the checks.

LESSON 25: Formalize your communication with ChatGPT with initial training. That way the communication can hopefully be way more efficient and less ambiguous.

Is ChatGPT just another tool?

- Google Search
- Stack Overflow
  - **Who Answers It Better? An In-Depth Analysis of ChatGPT and Stack Overflow Answers to Software Engineering Questions.**
  - More than 50% of ChatGPT's answers contain inaccuracies.
  - Scenario where SO shines.
  - In my experience, ChatGPT is good at solving problems in own code.
- GitHub Copilot
  - **Research: quantifying GitHub Copilot's impact on developer productivity and happiness.**
  - Devs using CoPilot are more than 2 x faster than devs not using CoPilot.
  - IDE vs Browser.
  - Code completion/pull requests vs broad managin code, tests, bugs.
- McKinsey
  - **Unleashing developer productivity with generative AI.**
  - Devs up to twice as fast with generative AI.
    - Expediting manual and repetitive work.
    - Jump-starting the first draft of new code.
    - Accelerating updates to existing code.
    - Increasing developers' ability to tackle new challenges.

Google DeepMind: AlphaCode
Amazon: CodeWhisperer
Microsoft: Copilot
Google: Bard
Google: Gemini
OpenAI: ChatGPT5