ChatGPT - Experiences from a programming course!

Debasis Bhattacharya, JD, DBA
University of Hawaii Maui College
debasisb@hawaii.edu
maui.hawaii.edu/cybersecurity
1. Overview
2. Programming Courses
   a. JavaScript
   b. Database Design and Programming - SQL
   c. Server Side Programming - MongoDB, Express, NodeJS, APIs etc.
3. ChatGPT
   a. Final Exams 2022
   b. Programming Learning Assistant
   c. Final Exams 2023
4. Reflections
"Traditional programming courses have relied on students learning coding syntax from examples and other textbook sources. As students learn to write code, they become proficient in the syntax, logic, debugging and documenting their code. ChatGPT is a revolutionary method that allows students to generate new code, fix bugs and errors and assist in the writing and learning of programming languages and computational concepts.

This session provides an outline of the application of ChatGPT is a server-side programming class at the University of Hawaii Maui College in the Spring semester of 2023. This session will cover the following topics -
1. Overview of ChatGPT
2. Introduction to Conversational Model in a Programming Course
3. Techniques to learn programming using ChatGPT

The session will provide the participants with an overview of how ChatGPT can be used as an assistive tool in the classroom, with examples of how it has been used in a programming course to help students learn to code. The session will provide examples of how the risks of plagiarism and cheating can be minimized using ChatGPT. Finally, the session will provide the participant with code samples and examples about how this new tool can be included in their own classroom. While this session focuses on the application of ChatGPT on a programming course, the lessons from this session can be applied to other courses in various disciplines unrelated to programming or technology."
1. Programming Courses
   a. JavaScript
   b. Database Design and Programming - SQL
   c. Server Side Programming - MongoDB, Express, NodeJS, APIs etc.

2. Students have basic knowledge of programming

3. Students use various tools and open source code
   a. Brackets and/or VS code editor
   b. PowerShell, CMD, Terminal etc.
   c. Open Source Coding
      i. HTML, CSS, JavaScript, JSON, NodeJS, Express, EJS
      ii. APIs, MongoDB, Mongoose, GitHub
   d. Documentation and tutorials
      i. Source Code web site
      ii. W3Schools
      iii. Khan Academy
      iv. Udemy courses
   e. Online course with live Zoom session once a week
      i. Students get detailed instructions about setup and homework
   f. Two Term Exams and a Final Exam - total 50% of the grade
ChatGPT v3.5
Free for students
Started using in March 2023
Students learned how to write clear and concise prompts and converse
Understand code snippets and examples from class
Used to generate sample code for MongoDB and Mongoose
Students used ChatGPT for homework and final exam - but not to cut and paste code!

ChatGPT v4.0 - Plus
Useful for an Instructor
Paid service for $20/month
More powerful engine for generating code and explanations
Assists instruction by providing an online coding “assistant”
Helpful for generating sample code for students based on existing documentation
Questions were all in text format and provided clear and concise instructions for students to write or fix code.

ChatGPT answered all questions with correct/elegant answers!
Question 1 – 4 points – Respond with answers in a Word Document

Consider the following code that creates a Mongoose Schema:

```javascript
const postSchema = {
  title: {
    type: String,
    required: [true, 'Title of Blog is required']
  },
  content: {
    type: String,
    required: [true, 'Content of Blog is required']
  }
};
```

Extend the schema with the following four elements:

1. **Description** of type String but always in lowercase, required field
2. **PostDate** of type Date, with default of today’s date, not a required field
3. **Rating** of type Number, with a min rating of 1, not a required field
4. **isValid** is a Boolean, required field
const postSchema = {
  title: {
    type: String,
    required: [true, 'Title of Blog is required']
  },
  content: {
    type: String,
    required: [true, 'Content of Blog is required']
  },
  Description: {
    type: String,
    lowercase: [true],
    required: [true],
    default: [Date.now]
  },
  PostDate: {
    type: Date,
  },
  Rating: {
    type: Number,
    min: [1]
  },
  isValid: {
    type: Boolean,
    required: [true, "Need to enter True or False"]
  }
};
const mongoose = require('mongoose');
const { Schema } = mongoose;

const postSchema = new Schema({
  title: {
    type: String,
    required: [true, 'Title of Blog is required']
  },
  content: {
    type: String,
    required: [true, 'Content of Blog is required']
  },
  description: {
    type: String,
    required: [true, 'Description is required'],
    set: (value) => value.toLowerCase()
  },
  postDate: {
    type: Date,
    default: Date.now,
  },
  rating: {
    type: Number,
    min: [1, 'Rating must be at least 1']
  },
  isValid: {
    type: Boolean,
    required: [true, 'isValid field is required']
  }
});
Write the HTML code for a simple form that requires a mandatory text input fields for username mandatory, a mandatory password type input field, and an optional text input type of Last Name. Ensure you have a placeholder with hints for every field and require input for uid and pwd.

```html
<form>
  <label for="username">Username:</label><br>
  <input type="text" id="username" name="username" placeholder = "Please enter your username" required><br>

  <label for="pwd">Password:</label><br>
  <input type="password" id="pwd" name="pwd" placeholder = "Please enter a valid password" required>

  <label for="lastname">LastName:</label><br>
  <input type="text" id="lastname" name="lastname" placeholder = "Please enter your last name" >
</form>
```
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Simple Form</title>
</head>
<body>
  <form>
    <label for="username">Username:</label>
    <input type="text" id="username" name="username" placeholder="Enter your username" required>
    <br><br>
    <label for="password">Password:</label>
    <input type="password" id="password" name="password" placeholder="Enter your password" required>
    <br><br>
    <label for="lastName">Last Name:</label>
    <input type="text" id="lastName" name="lastName" placeholder="Enter your last name (optional)">
    <br><br>
    <input type="submit" value="Submit">
  </form>
</body>
</html>
```
Introducing ChatGPT

We’ve trained a model called ChatGPT which interacts in a conversational way. The dialogue format makes it possible for ChatGPT to answer followup questions, admit its mistakes, challenge incorrect premises, and reject inappropriate requests.
GPT-3.5 (free)

GPT-4 (latest, $20/mo)
Type in a Prompt -

How Do I Install MongoDB on Windows

1. Download MongoDB:
   Visit the MongoDB official website’s download page. Select the “Server” option, and then choose the version you want (usually the current release), your platform (Windows), and package (MSI). Click the “Download” button.

2. Run the Installer:
   After the .msi file is downloaded, run it. This will start the installation process.

3. Install MongoDB:
   * Choose the “Complete” setup type to install MongoDB and all the required components.
   * Choose the “Custom” setup type if you want to specify any custom settings or if you want to install MongoDB at a particular location.

4. Service Configuration:
   * Choose “Run service as Network Service user” (default setting). This allows MongoDB to run as a service in Windows, starting automatically when the system starts.
   * Specify the data directory, which is where MongoDB stores its data. The default location is C:\data\db. If you choose another location, make sure you remember it as you’ll need it later.
   * Specify the log directory, which is where MongoDB will write logs. The default location is C:\Program Files\MongoDB\Server(version)\Log. If you choose another location, remember it for later.

5. Install MongoDB Compass (Optional):
   MongoDB Compass is the GUI for MongoDB. It’s not mandatory to install it, but it’s very useful for managing your database visually. Choose whether to install it based on your needs.

6. Finish the Installation:
   After you’ve made all your selections, click the “Install” button to install MongoDB. Once the installation is complete, click the “Finish” button.

After installation, MongoDB should be running as a service. You can access it via the MongoDB Shell or connect.
Prompt -

How Do I Install MongoDB Shell on Windows

(Note: training ended on Sept. 2021)
const blogSchema = new Schema(
  title: String, // String is shorthand for {type: String}
  author: String,
  body: String,
  comments: [{ body: String, date: Date }],
  date: { type: Date, default: Date.now },
  hidden: Boolean,
  meta: {
    votes: Number,
    favs: Number
  }
);

Your Mongoose schema appears to be well-structured and follows the standard practices for defining Mongoose schemas. Here is a review of each part:

1. **Importing Mongoose and Schema:** This is the standard way to import Mongoose and its Schema object. You've done this correctly.

2. **Defining the blogSchema:**
   - **title, author, body:** These are defined as `String` types. This shorthand is correct and commonly used in Mongoose schemas.
   - **comments:** This is an array of objects, each with a `body` of type `String` and a `date` of type `Date`. This is a good way to handle comments as subdocuments within the blog post document.
   - **date:** This is a `Date` type with a default value of `Date.now`. This means that if no date is provided when a new document is created, the current date and time will be used. This is a common practice for 'created at' fields.
   - **hidden:** This is a `Boolean` type. It could be used to hide or show the blog post based on its value.
   - **meta:** This is an embedded object with `votes` and `favs` properties, both of which are `Number` types. This is a good way to group related fields together.

Overall, your schema looks good and should work for a typical blog post structure. A few suggestions that could improve your schema:
Extend this Schema with Property Reviewer and Property Description
Open book exam for online students, live on Zoom

Students were allowed to prepare for the exam with ChatGPT or any other learning tool.

Students had access to ChatGPT during the exam. They were specifically instructed not to cut and paste from ChatGPT output.

Exam questions were based on images, minimum text, exam was a protected, non-editable PDF file...
Q1. Assume you have created the following documents in a MongoDB collection `inventory`.

```javascript
db.inventory.insertMany([  
  { item: "canvas", qty: 100, size: { h: 28, w: 35.5, uom: "cm" }, status: "A" },  
  { item: "journal", qty: 25, size: { h: 14, w: 21, uom: "cm" }, status: "A" },  
  { item: "mat", qty: 85, size: { h: 27.9, w: 35.5, uom: "cm" }, status: "A" },  
  { item: "mousepad", qty: 25, size: { h: 19, w: 22.85, uom: "cm" }, status: "P" },  
  { item: "paper", qty: 100, size: { h: 8.5, w: 11, uom: "in" }, status: "D" },  
  { item: "planner", qty: 75, size: { h: 22.85, w: 30, uom: "cm" }, status: "D" },  
  { item: "postcard", qty: 45, size: { h: 10, w: 15.25, uom: "cm" }, status: "D" },  
  { item: "sketchbook", qty: 80, size: { h: 14, w: 21, uom: "cm" }, status: "A" },  
  { item: "sketch pad", qty: 95, size: { h: 22.85, w: 30.5, uom: "cm" }, status: "A" }
]);
```

Write out the commands to query the following information using a MongoDB shell.

Write the code for Q1 and number your answers correctly.

1. Find the documents and the item names with the quantity greater than 50.
2. Update all the documents where the unit of measure is inches to centimeters.
3. Delete all the documents where the item quantity less than 45.
4. Find all documents where the item size height ranges from 10 to 14 inches.
Q2. Assume you have the following Mongoose schema called `breakfastSchema`:

```javascript
const breakfastSchema = new Schema({
  eggs: {
    type: Number,
    min: [6, 'Too few eggs'],
    max: 12
  },
  bacon: {
    type: Number,
    required: [true, 'Why no bacon?']
  },
  drink: {
    type: String,
    enum: ['Coffee', 'Tea'],
    required: function() {
      return this.bacon > 3;
    }
  }
});
const Breakfast = db.model('Breakfast', breakfastSchema);
```

Rewrite the entire `breakfastSchema` to include these new and updated criteria:

1. Eggs are required but the minimum is 2 eggs and maximum is 3.
2. Drinks could also include Orange Juice and Milk.
3. Bacon strips are required but only more than 2 strips.
4. Add a string called Toast, which is not required but could include White or Wheat toast.
Questions?
Comments?!
Debasis Bhattacharya
debasisb@hawaii.edu