Name: Justin Cabral CS509 Final Exam F22

|  |  |  |
| --- | --- | --- |
| Q1 |  | **20** |
| Q2 |  | **20** |
| Q3 |  | **20** |
| Q4 |  | **20** |
| Q5 |  | **20** |
| TOT |  | **100** |



**Q1 [20 pts.] Write the corresponding Use Case for the following proposed change to the Inventory Management System.**

*A corporate user would like to know the store with the greatest inventory value.*

**Use Case: Show Store with Greatest Inventory Value**

Participating Actor: Initiated by Corporate User

Entry Condition: Must be logged in as a Corporate User

Exit Criteria: Store with the greatest inventory value is shown

Flow of events:

1. Corporate user requests the store with the greatest inventory value
2. Inventory Management System retrieves the store with the greatest inventory value, shows store, and refreshes display

**Q2 [20 pts.]** Given the following application domain, construct a UML class analysis diagram (including attributes and relationships between classes) for the following context:

*You’ve been asked to run an auction website where users can register an account using an email and a password. Each of the items for auction has a name and an initial starting price. A user can place a bid on an item with a price that is greater than all existing bids recorded for that item. When the auction ends for an item, the website must record the winning bid (and user) for that item.*

**Show attributes and relationships between classes. Do not show methods or constructors**

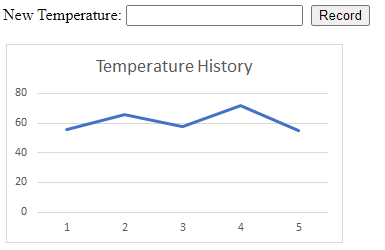
Diagram

Description automatically generated

**Q3 [20 pts.]** Given the following problem, show the UML class diagram for the appropriate React **entity**, **boundary** and **controller** classes that you would use. This application has one use case:

* Record Temperature

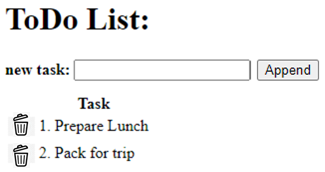
Recorded temperatures are graphed over time using a simple line chart. The chart is redrawn after a new temperature is recorded.



There is no need to show relationships (i.e., arrows) between the classes. Use proper UML class diagrams to capture the attributes and methods.

A picture containing diagram

Description automatically generated

**Q4 [20 pts.]** You want to maintain an enumerated list of tasks. You have identified the following use cases and storyboard:

* Show Tasks
* Append Task
* Remove Task

Describe the HTTP resource entry points, including:

• JSON payload sent (if needed) for each entry point

• Status Code(s) in response

• JSON response (if needed) for each entry point

**Show Tasks**

/show

GET (no payload required)

200 Success:

TASKS

400 Failure:

{ “error” : “No Tasks” }

**Append Task**

/append\_task

POST

{ “task” : “Buy toothpaste”}

200 Success:

TASKS

400 Failure:

{ “error” : “Couldn’t append task” }

**Remove Task**

/remove\_task

POST

{ “task” : “Buy toothpaste”}

200 Success:

{ “success”: “Task was removed successfully” }

400 Failure:

{ “error” : “Could not remove task” }

{ “error” : “Task does not exist” }

**Payload Definitions**

**TASKS**

{ “tasks” : [

{ “task” : “Prepare lunch” },

{ “task” : “Pack for trip” },

…

]

}

**Q5 [20 pts.]** These questions are concerned with AWS. There are SIX possible questions, and you only need to answer **FOUR**. Note that I will only grade the first FOUR of your answers, so do not answer more than FOUR. **Identify the FOUR questions that you have chosen to answer.**

1. Describe the structure of the input to a JavaScript lambda function.
2. Describe how you were able to host your React (or straight HTML and JavaScript) application using a URL provided by AWS.
3. If you used Cloud9 for your code development, explain what “**sam local invoke**” allowed you to do.
4. Describe how you can validate that a POST resource entry point within API Gateway is working as expected.
5. Describe how your React (or straight HTML and JavaScript) application communicates to your REST-based API hosted on AWS.
6. Describe how your lambda function retrieves information from the RDS.
7. The structure of input to a JavaScript lambda function is a HTTP response which can either be GET, POST, PUT, or DELETE to the lambda endpoint. Within those responses there can be an event body (body is empty with GET) containing a JSON payload which can then be parsed by the lambda function.
8. I was able to host the React App using the URL provided by AWS by using the command npm build which created a build folder inside my React project directory. I would then take the contents of that build folder and upload them to the AWS S3 bucket. This would then create a static website with the URL I could share.

E.) The way my react application communicates with the REST-based API hosted on AWS is that it uses the fetch() function in JavaScript to send HTTP requests to the API endpoints. To get data from those endpoints, I can make requests such as GET and POST which would cause the API to return the data in a response. I also had to make sure that the proper CORS policies were in place such as allowing the proper domains the make requests to the API.

F.) To retrieve information from the RDS, my lambda function can create a connection to the RDS database by providing the lambda function a RDS Endpoint URL, username, password, as well as the schema you are looking to query. Once those obligations are met, the queries can be done on the database and we can wait for the response which will be returned.