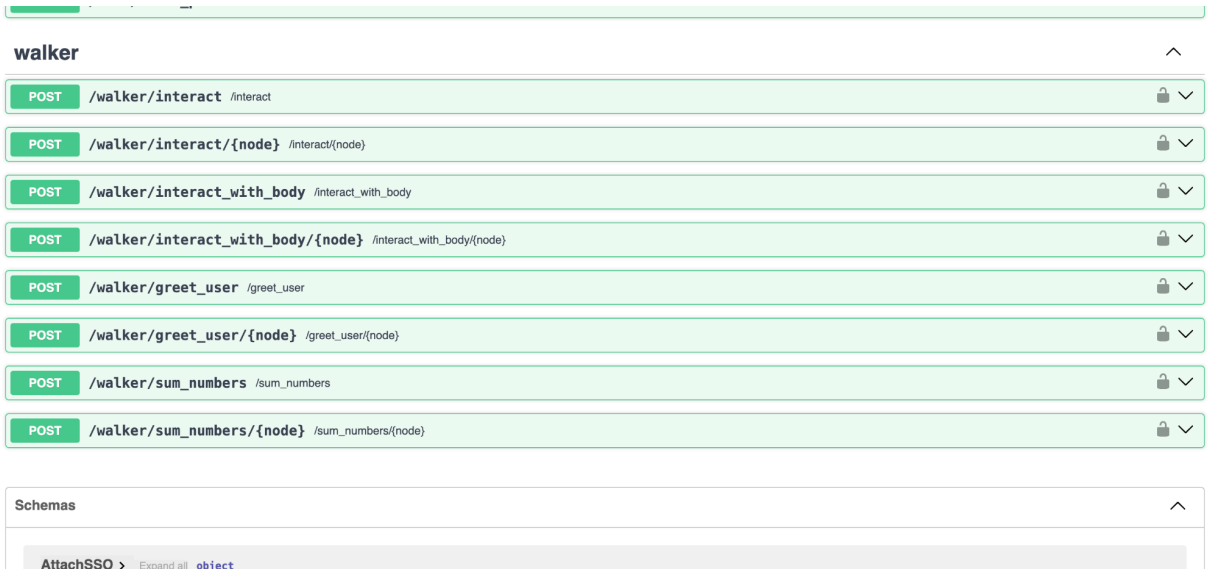
How things went:

During this assignment, I encountered various challenges while implementing the custom state, routing, and the chat node, particularly with setting up MongoDB and Docker. Configuring the MongoDB replica set and getting Docker to run correctly was tricky, and I had to spend time troubleshooting these issues. The token-based authentication with the API also posed some difficulties as I worked through multiple curl commands, debugging issues with environment variables and ensuring the correct server responses. This process taught me the importance of persistence and troubleshooting, as even small misconfigurations could lead to errors in the chatbot's functionality. Setting up JWT tokens for authentication and handling interactions with the backend was a key learning experience, giving me a better understanding of how AI-driven applications interact with server environments.

Additionally, I faced challenges in integrating the custom ChatType state into the chatbot and ensuring it routed queries properly. Working with server logs and troubleshooting real-time issues helped me develop a more systematic approach to debugging. Throughout this process, I learned valuable lessons about configuring backend systems, working with containers, and how critical it is to carefully test each stage of the development pipeline to avoid larger issues down the road. The experience highlighted the importance of patience and meticulous testing when implementing complex backend systems.

Part 1:



```
curl: (3) URL using bad/illegal format or missing URL
rithvikvanga@Rithviks-MacBook-Air ~ % curl -X POST http://localhost:8000/walker/gre
et_user \
-H "Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY2Zjc0MjY5
NzIwYzU5NDgyMmUyYjBlZCIsImVtYWlsIjoidGVzdEBtYWlsLmNvbSIsInJvb3RfaWQiOiI2NmY3NDI2OTc
yMGM1OTQ4MjJlMmIwZWMiLCJpc19hY3RpdmF0ZWQiOnRydWUsImV4cGlyYXRpb24iOjE3Mjc1MjQ2NTUsIn
N0YXRlIjoiSWZrcTJMSk0ifQ.zXbnj-8DRNZm4vpojIvzrrAVYjg6Uy4mIwv_xz3RoIQ" \
-H "Content-Type: application/json" \
-d '{"name": "Rithvik"}'

{"status":200,"reports":[{"response":"Hello, Rithvik!"}]}%
rithvikvanga@Rithviks-MacBook-Air ~ %
```

```
rithvikvanga@Rithviks-MacBook-Air ~ % curl -X POST http://localhost:8000/walker/sum
_numbers \
-H "Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY2Zjc0MjY5
NzIwYzU5NDgyMmUyYjBlZCIsImVtYWlsIjoidGVzdEBtYWlsLmNvbSIsInJvb3RfaWQiOiI2NmY3NDI2OTc
yMGM1OTQ4MjJlMmIwZWMiLCJpc19hY3RpdmF0ZWQiOnRydWUsImV4cGlyYXRpb24iOjE3Mjc1MjUxNzYsIn
N0YXRlIjoiVTkyQzJGSEkifQ.33G0pw6y3Jtfk2yH7s5xldU6qciTZbMCbSXlJHLz49c" \
-H "Content-Type: application/json" \
-d '{"num1": 5, "num2": 7}'

{"status":200,"reports":[{"response":"The sum is 12."}]}%
rithvikvanga@Rithviks-MacBook-Air ~ %
```

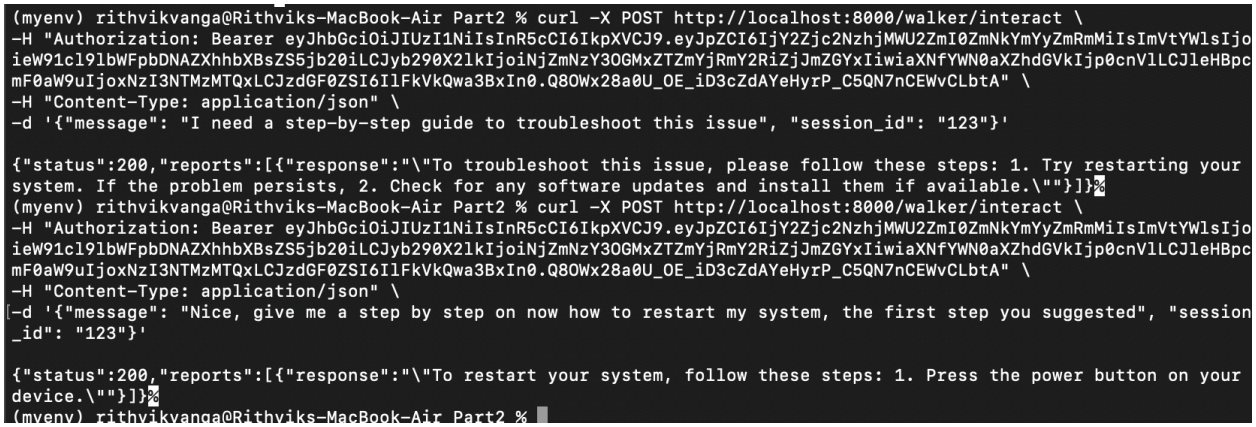
Part 2:(2 diff models)

```
(myenv) rithvikvanga@Rithviks-MacBook-Air Part2 % curl -X POST http://localhost:800
0/walker/interact \
-H "Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY2Zjc2MTFj
YWMwNmNiNGJkMTU2NjBkNiIsImVtYWlsIjoieW91cl9lbWFpbDJAZXhhbXBsZS5jb20iLCJyb290X2lkIjo
iNjZmNzYxMWNhYzA2Y2I0YmQxNTY2MGQ1IiwiaXNfYWN0aXZhdGVkIjp0cnVlLCJleHBpcmF0aW9uIjoxNz
I3NTMxNDkzLCJzdGF0ZSI6Im9vVG1qbENIIn0.Zyo6vwK8l_RimEyFiu6Npara2AMMHyBEq0ht74DfUhE"
\
-H "Content-Type: application/json" \
-d '{"message": "Tell me about ridge regression", "session_id": "123"}'

{"status":200,"reports":[{"response":"The function h(x;θ) is a sigmoidal threshold
function with parameters (k, position, θ0, direction, θ1). This function is used in
 a single layer neural network trained using SGD.\n\nThe learning rule for this net
work is:\nθ(k+1)=θ(k)−ηk∇Loss(y(i)h(x(i);θ)), where z=h(¯x(i);¯θ), ∂Loss (yz) / ∂wi
 = −yxj, and the weights w(k+1) are updated accordingly.\n\nFor a two-layer neural
network using backpropagation, you can use GD with the chain rule. The learning rul
e is:\nv(k+1) = v(k) − ηkyhj[[(1−yz)>0]], where ∂Loss(yz) / ∂wji = ∂Loss(yz) / ∂z∂z
 ∗ ∂hj∂hj ∗ ∂zj∂zj ∗ ∂wji, and the weights w(k+1) are updated accordingly.\n\nThe a
ctivation functions for this neural network are: rectified linear (f(z) = max(0, z)
), threshold (f(z) = sign(z)), sigmoid (f(x) = 1 / (1 + e−x)), and tanh (f(z) = tan
h(z)).\n\nIn Bayesian Networks, the log-likelihood is used for MLE learning. The co
rresponding MLE is: ˆθi(xi|xpai) = #(Xi=xi, Xpai=xpai) / ∑x'i #(Xi=x′i, Xpai=xpai),
 where # denotes the count of an event.\n\nThe Bayesian Information Criterion (BIC)
 is used for model selection: BIC(D;θ) = l(D;θ) − # param ∗ 2log(n). The Mth order
Markov Model uses this criterion for learning."}]}%
(myenv) rithvikvanga@Rithviks-MacBook-Air Part2 %
```

```
{"status":200,"reports":[{"response":"1. To find the degree of a vertex (di), sum up the weights of all edges connecte
d to that vertex. This gives us the number of connections each vertex has.\n\n2. The degree matrix (D) consists of the
 degrees of all vertices, where Di,i is the degree of vertex i and Dij = 0 for i≠j.\n\n3. The cost of a cut between se
t A and A is the total weight of the edges that separate this set from the rest of the network.\n\n4. The Graph Laplac
ian (L) is calculated by subtracting the degree matrix (D) from the weight matrix (W). It's used to analyze network pr
operties.\n\n5. Creating a matrix with the first k eigenvectors as columns gives us a reduced representation of the da
ta, where each row can be interpreted as a new data point in a k-dimensional space.\n\n6. Applying k-means to the new
 data representation involves grouping similar points into clusters based on their proximity, with the number of cluste
rs specified by k.\n\n7. Hierarchical clustering is a method for finding groups within data and creating a hierarchy t
hat shows how the groups are related to each other, by merging closest clusters recursively until convergence or a pre
defined number of clusters remain.\n\n8. In recommender systems, Nearest Neighbor Prediction uses one user's ratings t
o predict the rating for missing items. It calculates the predicted rating based on similarity between users and their
 common movie ratings.\n\n9. The Backpropagation algorithm trains a neural network by adjusting its weights to minimiz
e the error between predicted and actual output values, through iteratively updating weights based on the gradient of
the loss function.\n\n10. The Gradient is a vector containing the partial derivatives of a scalar-valued function at s
ome point in space. In multivariable calculus, gradients can be combined using rules such as product rule and chain ru
le.\n\n11. The Hessian matrix contains the second-order partial derivatives of a scalar-valued function and describes
the local curvature at a point in space.\n\n12. Eigenvalues (λ) and eigenvectors (⃗ v) are solutions to an equation inv
olving a square matrix A, where λ is the eigenvalue and⃗ v is the eigenvector that satisfies (A − λI) ∗⃗ v = 0 for som
e scalar value λ. The set of eigenvectors forms a basis for the space in which A operates, and their associated eigenv
alues determine how much each eigenvector is stretched or squashed by the matrix A.\n\n13. Positive (semi-)definite ma
trices have properties where xT A x > 0 for all nonzero x (positive definite) or xT A x ≥ 0 for all x (positive semi-d
efinite). In other words, they ensure that the matrix multiplica(myenv) rithvikvanga@Rithvik(myenv) rithvikvanga@Rithv
ik(myenv) rithvikvanga@Rithvik(myenv) rithvikvanga@Rithvik(myenv) rithvikvanga@Rithvik(myenv) rithvikvanga@Rithvik(mye
nv) rithvikvanga@Rithvik(myenv) rithvikvanga@Rithvik(myenv) rithvikvanga@Rithvik(myenv) rithvikvanga@Rithvik(myenv) ri
thvikvanga@Rithvik(myenv) rithvikvanga@Rithvik(myenv) rithvikvanga@Rithvik(myenv) rithvikvanga@Rithvik(myenv) rithvikv
anga@Rithvik(myenv) rithvikvanga@Rithvik(myenv) rithvikvanga@Rithvik(myenv) rithvikvanga@Rithvik(myenv) rithvikvanga@R
ithvik(myenv) rithvikvanga@Rithvik(myenv) rithvikvanga@Rithvik(myenv) rithvikvanga@Rithvik(myenv) rithvikvanga@Rithvik
```

Part 3:



The GUIDE state in the ChatType enum represents a scenario where the user is requesting detailed, step-by-step guidance or troubleshooting assistance. This state is used to classify queries where the user might be asking for instructions on how to complete a task or resolve an issue. In the chatbot, this state enables a more structured and instructional response, rather than a simple answer or a document-based response. For example, if a user asks, "How do I reset my password?" the chatbot would recognize this as a request for a guide and respond with a step-by-step process to help the user complete the task.

The chatbot uses a Router node to classify incoming user queries. When the user submits a query, the Router classifies it based on the message's content. If the query is determined to be a request for a step-by-step guide, it is classified under the GUIDE state. The chatbot will then route the query to the GuideChat node, which is specifically designed to handle such requests. Once routed, the GuideChat node generates a response with detailed, step-by-step instructions, tailored to the user’s query. This ensures that the chatbot provides the most appropriate type of response based on the classification.