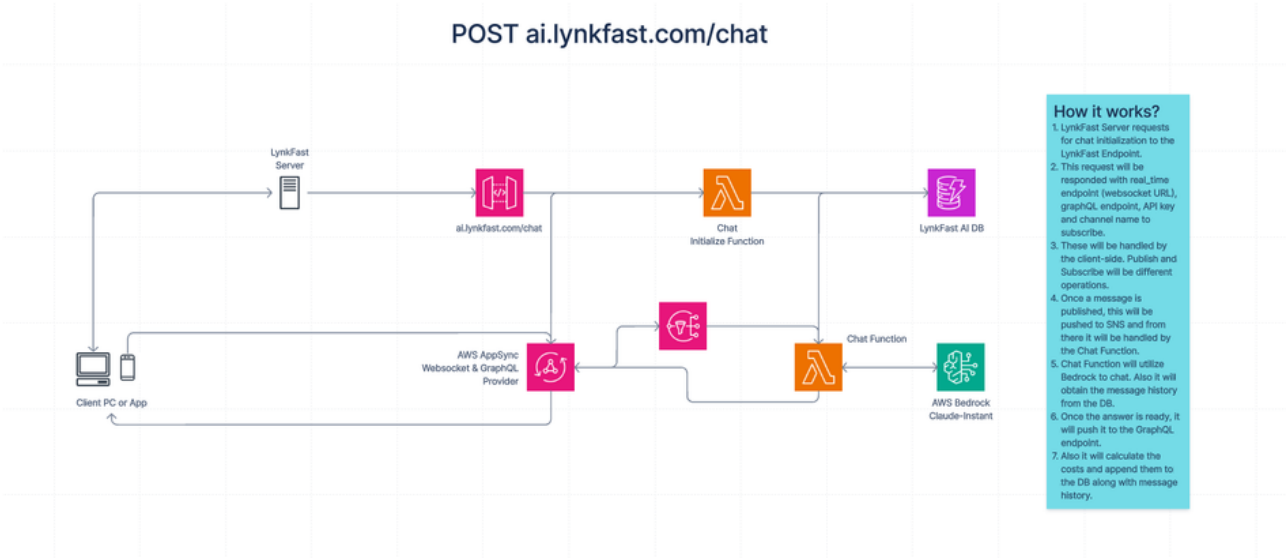
Chat Documentation

Chat Architecture



POST /chat

This function initiates the communication between AI model and Client.

Header

`x-api-key` : `<ai_api_key>`

Payload

```
{
    "project_name":"<string>", (Job Name)
    "setup":{
        "type_of_work":"<string>",
        "job_type":"<string>",
        "booking_date":"<datetime format>",
        "modified_date":"<datetime format> | NONE", (OPTIONAL)
        "material_provided":"CLIENT | COMPANY"
    },
    "customer":{
        "name":"<string>", (OPTIONAL)
        "address":"<string>", (Contains Address Lines, City, State and Zip)
        "contact_person":"<string>",
        "phone_number":"<string>",
        "email":"<string>" (OPTIONAL)
    },
    "job_details":{
        "address":"<string>", (Contains Address Lines, City, State and Zip)
        "contact_person":"<string>", (OPTIONAL)
        "phone_number":"<string>",(OPTIONAL)
        "email":"<string>"(OPTIONAL),
        "job_description":"<string>",
        "project_status":"<string>"
    },
```

```
    "gc":{ (OPTIONAL)
        "name":"<string>", (OPTIONAL)
        "address":"<string>", (Contains Address Lines, City, State and Zip)
        "contact_person":"<string>",
        "phone_number":"<string>",
        "email":"<string>" (OPTIONAL)
    },
    "building_owner":{ (OPTIONAL)
        "name":"<string>", (OPTIONAL)
        "address":"<string>", (Contains Address Lines, City, State and Zip)
        "contact_person":"<string>",
        "phone_number":"<string>",
        "email":"<string>" (OPTIONAL)
    },
    "team_members":[ (Can be multiple)
        {
            "name":"<string>",
            "role":"<string>",
            "hourly_rate":"<float>",
            "certifications":"<string>",
            "description":"<string>"
        }
    ],
    "technical_details":[
        {
            "page_name":"<string>",
            "parent_page":"<string> | NONE", (Only HomePage can be NONE)
            "layers":[ (Assets on this page)
                {
                    "asset_label":"<string>",
                    "location":"<string>",
                    "size":"<width X height>",
                    "title":"<string>",
                    "connection_count":"<int>",
                    "termination_count":"<int>",
                    "connections":[
                        {
                            "start_port":"<string>",
                            "start_device":"<string> | NONE",
                            "end_port":"<string>",
                            "end_device":"<string> | NONE"
                        }
                    ],
                    "kit_items":[
                        {
                            "kit_item_name":"<string>",
                            "quantity":"<int>",
                            "wire_lenght":"<int>", (OPTIONAL)
                            "measure_unit":"<string>",
                            "install_time":"<int>", (Format would be in minutes)
                            "description":"<string>",
                            "upc":"<string>",
                            "manufacturer_name":"<string>",
                            "status":"<string>"
                        }
                    ]
                }
            ]
```

```
        }
    ],
    "bom":[
        {
            "upc":"<string>",
            "name":"<string>",
            "quantity":"<int>",
            "total_installation_time":"<int>", (Format would be in minutes)
            "labor_price":"<float>",
            "total_price":"<float>"
        }
    ],
    "project_timer":"<int>", (In Minutes)
    "tasks":[
        {
            "item":"<string>",
            "location":"<string>",
            "category":"<string>",
            "total_time_estimated":"<int>", (In Minutes)
            "total_time_used":"<int>", (In Minutes)
            "status":"<string>"
        }
    ],
    "billing_information":{
        "lien_required":"<bool>",
        "payment_terms":"<string>",
        "po_number":"<string>",
        "contract_number":"<string>",
        "address":"<string>", (Contains Address Lines, City, State and Zip)
        "contact_person":"<string>",
        "billing_type":"<string>",
        "contract_type":"<string>",
        "sales_tax":"<bool>",
        "contract_amount":"<float>",
        "insuarance_required":<bool>,
        "certified_payroll":"<bool>",
        "prevailing_wage":"<bool>",
        "performance_bond":"<bool>",
        "warranty_offered":"<bool>",
        "manufacture":"<string>",
        "retention_witheld":"<string>",
        "type":"<string>",
        "representetive":"<string>",
        "estimator":"<string>",
        "project_manager":"<string>"
    },
    "proposal_documents":[
      {
        "title":"<string>",
        "description":"<string | base64>" (NOT NULL)
      }
    ]
}
```

Possible Responses

Status Code	Message	Note

412	Function Initialization Error	Check Your Payload
515	Prompt Preparation Error	Contact Developer
512	DynamoDB Error	Contact Developer
516	AppSync Error	Contact Developer

200	Successful	<pre>{
		  "channel_name":"<unique channel name>"
		  "realtime_endpoint":"wss://<url>",
		  "request_endpoint":"https://<url>",
		  "api_key":"<string>"
		}</pre>

After starting the Chat you need to first subscribe to the channel using the `realtime_endpoint` and `api_key`. After this, please send the message using the `request_endpoint` & `api_key`.

Subscribe to Channel (GET <realtime_endpoint>)

This payload allows you to subscribe to the messages.

Payload

```
{
  "id":"<random_generated_id>",
  "payload":{
    "data":"{"query":"<string graphQL query>"}", (Whole data will be string)
    "extensions":{
      "authorization":{
        "x-api-key":"<api_key>"
      }
    }
  },
  "type":"start"
}
```

The GraphQL Query will be like this:

```
subscription SubscribeToData {
  subscribe(name:"<channel_name>") {
    name,
    data
  }
}
```

If successful, You should receive a message for `start_ack`.

Incoming Message Structure

```
{
  "data": {
    "subscribe": {
      "name": "<channel_name>",
```

```
      "data": "<message | base64>"
    }
  }
}
```

Publish Message to Channel (POST <request_endpoint>)

This payload is for input messages to the chat as the user.

Payload

```
{
  "query":"<string | graphQL query>"
}
```

The GraphQL Query will be like this:

```
mutation sendmessage {
  client_publish(message: "{\"channel\":\"<channel_name>\",\"message\":\"<string_message>\"}")
}
```

Response

```
{
  "data": {
    "client_publish": <boolean>
  }
}
```

⚠ If the `boolean` is `True` it means that message is sent to the queue. But it doesn't necessarily mean the payload is correct. Since we have async architecture here, message gets denied if the payload is not correct, and no error response returns. Only if you receive response back, that's when you can be sure that you have correct structure.

ℹ If the `boolean` is `False` there is something fundamentally wrong, contact the developer.