3. Create a Bedrock Service Class

Create a service class to handle Bedrock agent interactions:

```php
<?php

namespace App\Services;

use Aws\BedrockAgentRuntime\BedrockAgentRuntimeClient;
use Aws\Exception\AwsException;
use Illuminate\Support\Facades\Log;

class BedrockAgentService
{
    private $client;
    private $agentId;
    private $agentAliasId;

    public function __construct()
    {
        $this->client = new BedrockAgentRuntimeClient([
            'version' => 'latest',
            'region' => config('aws.region', 'us-east-1'),
            'credentials' => [
                'key' => config('aws.access_key_id'),
                'secret' => config('aws.secret_access_key'),
            ]
        ]);

        // Your agent details from the URL: 5ZBEQIUR3J
        $this->agentId = '5ZBEQIUR3J';
        $this->agentAliasId = 'TSTALIASID'; // Replace with your actual alias ID
    }

    public function invokeAgent($prompt, $sessionId = null)
    {
        try {
            // Generate session ID if not provided
            if (!$sessionId) {
                $sessionId = uniqid('session_', true);
            }

            $result = $this->client->invokeAgent([
                'agentId' => $this->agentId,
                'agentAliasId' => $this->agentAliasId,
                'sessionId' => $sessionId,
                'inputText' => $prompt,
            ]);

            // Process the streaming response
            $completion = '';
            $eventStream = $result->get('completion');

            foreach ($eventStream as $event) {
                if (isset($event['chunk'])) {
                    $chunk = $event['chunk'];
                    if (isset($chunk['bytes'])) {
                        $completion .= $chunk['bytes'];
                    }
                }
            }

            return [
                'success' => true,
                'response' => $completion,
                'sessionId' => $sessionId
            ];

        } catch (AwsException $e) {
            Log::error('Bedrock Agent Error: ' . $e->getMessage());
            return [
                'success' => false,
                'error' => $e->getMessage(),
                'sessionId' => $sessionId
            ];
        }
    }
}
```

4. Create a Controller

Create a controller to handle the agent requests:

```
<?php

```

```php
namespace App\Http\Controllers;

use App\Services\BedrockAgentService;
use Illuminate\Http\Request;
use Illuminate\Http\JsonResponse;

class BedrockAgentController extends Controller
{
    private $bedrockService;

    public function __construct(BedrockAgentService
$bedrockService)
    {
        $this->bedrockService = $bedrockService;
    }

    public function chat(Request $request):
JsonResponse
    {
        $request->validate([
            'message' => 'required|string|max:1000',
            'session_id' => 'nullable|string'
        ]);

        $response = $this->bedrockService-
>invokeAgent(
            $request->input('message'),
            $request->input('session_id')
        );

        return response()->json($response);
    }
}
```

5. Add Routes

Add routes to your `routes/web.php` or `routes/api.php`:

```
<?php

use App\Http\Controllers\BedrockAgentController;

// For API routes (routes/api.php)
Route::post('/bedrock/chat',
[BedrockAgentController::class, 'chat']);

// For web routes (routes/web.php)
Route::post('/bedrock/chat',
[BedrockAgentController::class, 'chat'])-
>name('bedrock.chat');
```

6. Frontend Integration (Blade Template)

Create a simple chat interface:

```
<!-- resources/views/chat.blade.php -->
<!DOCTYPE html>
<html>
<head>
    <title>Bedrock Agent Chat</title>
    <meta name="csrf-token"
content="{{ csrf_token() }}">
    <script src="https://code.jquery.com/
jquery-3.6.0.min.js"></script>
</head>
```

```
<body>
    <div id="chat-container">
        <div id="messages"></div>
        <div>
            <input type="text" id="message-input"
placeholder="Type your message...">
            <button onclick="sendMessage()">Send</
button>
        </div>
    </div>

    <script>
        let sessionId = null;

        function sendMessage() {
            const message =
document.getElementById('message-input').value;
            if (!message.trim()) return;

            // Add user message to chat
            addMessage('You: ' + message);
            document.getElementById('message-
input').value = '';

            // Send to Bedrock agent
            $.ajaxSetup({
                headers: {
                    'X-CSRF-TOKEN': $
('meta[name="csrf-token"]').attr('content')
                }
            });

            $.post('/bedrock/chat', {
                message: message,
                session_id: sessionId
            })
            .done(function(response) {
                if (response.success) {
                    addMessage('Agent: ' +
response.response);
                    sessionId =
response.sessionId; // Maintain session
                } else {
                    addMessage('Error: ' +
response.error);
                }
            })
            .fail(function() {
                addMessage('Error: Failed to connect
to agent');
            });
        }

        function addMessage(message) {
            const messagesDiv =
document.getElementById('messages');
            messagesDiv.innerHTML += '<div>' +
message + '</div>';
            messagesDiv.scrollTop =
messagesDiv.scrollHeight;
        }

        // Allow Enter key to send message
        document.getElementById('message-
input').addEventListener('keypress', function(e) {
            if (e.key === 'Enter') {
                sendMessage();
            }
        });
    </script>
</body>
</html>
```

7. Configuration File

Create a config file for AWS settings:

```
<?php
// config/aws.php

return [
    'access_key_id' => env('AWS_ACCESS_KEY_ID'),
    'secret_access_key' =>
env('AWS_SECRET_ACCESS_KEY'),
    'region' => env('AWS_DEFAULT_REGION', 'us-
east-1'),
];
```

8. Get Your Agent Alias ID

You'll need to get your actual agent alias ID. Run this command to find it:

```
aws bedrock-agent list-agent-aliases --agent-id
5ZBEQIUR3J --region us-east-1
```

Run in CloudShell

Or use this PHP code to get it programmatically:

```
<?php
```

```
use Aws\BedrockAgent\BedrockAgentClient;

$client = new BedrockAgentClient([
    'version' => 'latest',
    'region' => 'us-east-1',
    'credentials' => [
        'key' => env('AWS_ACCESS_KEY_ID'),
        'secret' => env('AWS_SECRET_ACCESS_KEY'),
    ]
]);

$result = $client->listAgentAliases([
    'agentId' => '5ZBEQIUR3J'
]);

foreach ($result['agentAliasSummaries'] as $alias) {
    echo "Alias ID: " . $alias['agentAliasId'] .
"\n";
    echo "Alias Name: " . $alias['agentAliasName'] .
"\n";
}
```

Usage

1. Update the `$agentAliasId` in the `BedrockAgentService` class with your actual alias ID
2. Make sure your AWS credentials have the necessary permissions (`bedrock:InvokeAgent`)
3. Test the integration by visiting your chat page and sending messages

This setup provides a complete integration of your Bedrock agent into your Laravel application with session management and error handling.