Sprint 3

GOAL: Route undecided requests to a physician so they can make a decision on the request

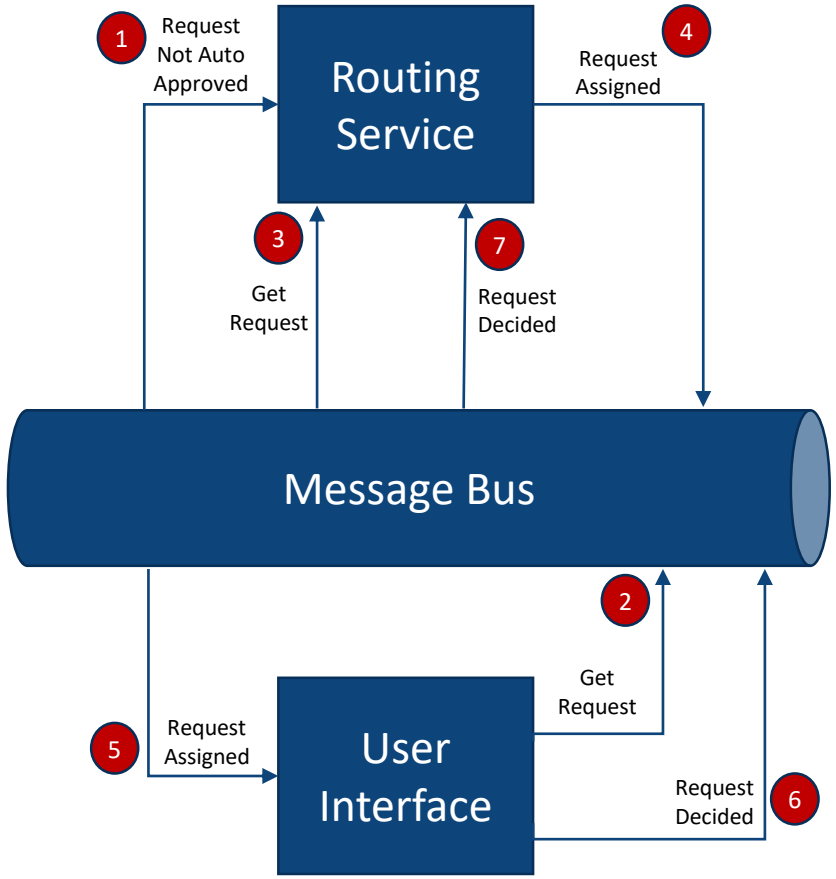
User Story 3-1

As a TAH physician, I need to be able to view undecided requests for service so that I can decide if they should be approved or denied.

Acceptance Criteria

1. Retrieve and view an undecided request without need to install an application
2. Should not be able to retrieve a request that has previously been decided
3. Should not be able to retrieve a request that has been assigned to another physician
4. Ability to approve or deny a request I’m viewing

User Story 3-1: Design



1. Routing Service consumes requests that were not auto approved
2. User asks for an undecided request to work on
3. Routing Service consumes the request to get a request for the user
4. Routing Service finds an undecided request, assigns it to the user, and sends an event for the assignment
5. UI consumes the assignment event and displays the request to the user
6. User makes a decision on the request and the UI sends an event for the decision
7. Routing Service consumes the decision event and removes the request from its database since it no longer needs to be routed

Side note: The Provider Transfer Service would also consume the decision event so it can notify the provider.

Routing Service

- Maintains database of routable requests
  - Routable requests include anything not already decided and not assigned to another user
  - No need to store anything already decided
- For this workshop, we're only going to store requests in memory
  - Note that this isn't possible with an Azure Function on the consumption plan
- How to host this functionality?

Azure Function vs App Service

	Azure Function (Consumption Plan)	Azure App Service
Cost	Cheaper for smaller, infrequent workloads. First 400,000 GB/s and 1,000,000 executions free.	Can be cheaper for larger, frequent workloads. Min for production app about $70/month without any use. Can be ***much*** higher.
Development Time	Simple to develop, debug, and deploy. Can all be done in portal, but not recommended.	Requires more time to develop, debug, configure, and deploy
Execution Time Limit	Generally 10 minutes, but 230 seconds for HTTP triggers	Unlimited
Scalability	Scales out well without configuration	Can be configured to scale out and/or up
Latency	Can take 10 – 20 seconds to warm up	Can be configured as “Always on” to eliminate cold start
Maintain State in Memory	Not possible	Technically possible when always on, but not recommended because it limits scaling out
General Use Case	Small, isolated, lightweight functionality	Larger set of cohesive functionality

User Interface

- Could use almost any web technology since there is a JavaScript Azure Event Hub Client SDK

- Choosing ASP.NET Blazor to consolidate technologies for the workshop

Development Approach

- Routing Service already developed and almost ready to deploy
  - Needs some configuration updates as defined in following tasks

- Use Azure Resource Manager (ARM) templates to generate the new event hubs, shared access policies, and BLOB containers for hub offsets

- We'll just use the UI locally without deploying

Azure Resource Manager (ARM)

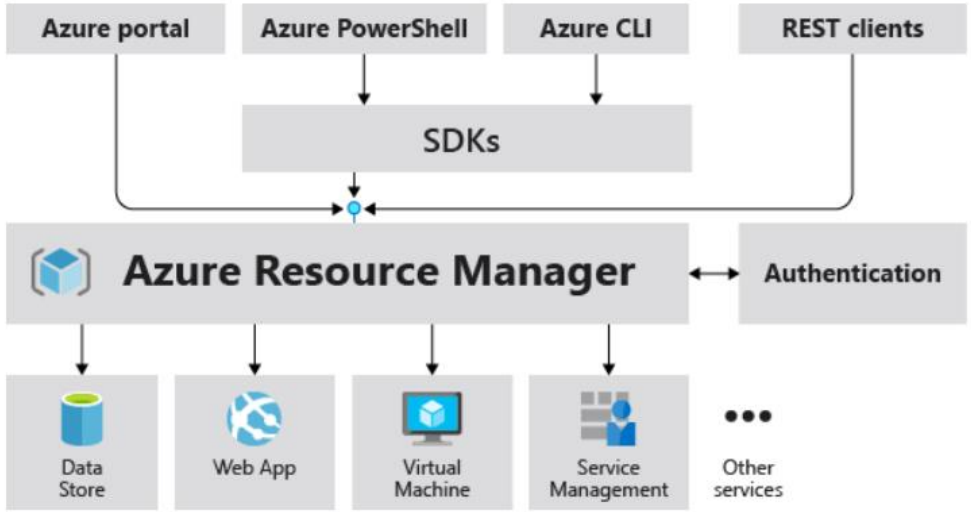
- Management layer that enables creating, updating, and deleting Azure resources

- We've been using directly via the Portal

- Now we're going to use ARM

Templates to create new resources

  - Infrastructure as Code (IaC)



Task 3-1: Deploy Routing Service Messaging Items

- BusinessLogic\Snippets\ArmScripts\Sprint3_1_RoutingServiceMessagingItems.json
  - ***Use the name of your Hub namespace***

- New hubs
  - getRequest
  - requestAssigned

- Share Access Policies
  - requestNotAutoApproved listener
  - decidedRequests listener
  - getRequest listener
  - requestAssigned sender

- Consumer Groups
  - requestNotAutoApproved
  - decidedRequests
  - getRequest

Task 3-2: Deploy UI Messaging Items

- BusinessLogic\Snippets\ArmScripts\Sprint3_2_UIMessagingItems.json
  - ***Use the name of your Hub namespace***

- Share Access Policies
  - getRequest sender
  - requestAssigned listener
  - decidedRequests sender

- Consumer Groups
  - requestAssigned

- There are other items in this script, but may not use

Task 3-3: Deploy Hub Offset Storage

- BusinessLogic\Snippets\ArmScripts\Sprint3_3_StorageItems.json
  - ***Set "Storage Account Name" to your unique name***

Component	Hub	Blob Container
Routing Service	notAutoApproved	*routingNotAutoApprovedConsumer*
Routing Service	requestDecided	*routingRequestDecidedConsumer*
Routing Service	getRequest	*routingGetRequestConsumer*
UI	requestAssigned	*uiRequestAssignedConsumer*

Task 3-4: Deploy Routing Service

- Update connection strings in the service
  - Top of RoutingService\Program.cs

- Publish Routing Service from Visual Studio
  - **Target**: Azure -> WebJobs
  - Use **+** to create new App Service
    - **Name** must be globally unique
    - Use same **Resource group** you've been using
    - Create new **Hosting Plan**
      - **Name** must be globally unique
      - **Size**: *Shared* to save money on features we don't need for the workshop
  - Must select **WebJob Type** = ***Continuous*** on publish summary page

Task 3-5: Create UI

- Update connection strings in UI\Variables.cs
  - Double check other variables are correct too

- Just run locally without deploying

Sprint 3: Retrospective

- When compared with Azure Functions, App Services are generally used for larger, heavier, and cohesive set of functionality, but comes with added cost and complexity
- ARM Templates can be used to create Azure resources
- Asynchronous messaging can be used for request/response
- Event Hubs can be used with a variety of technologies