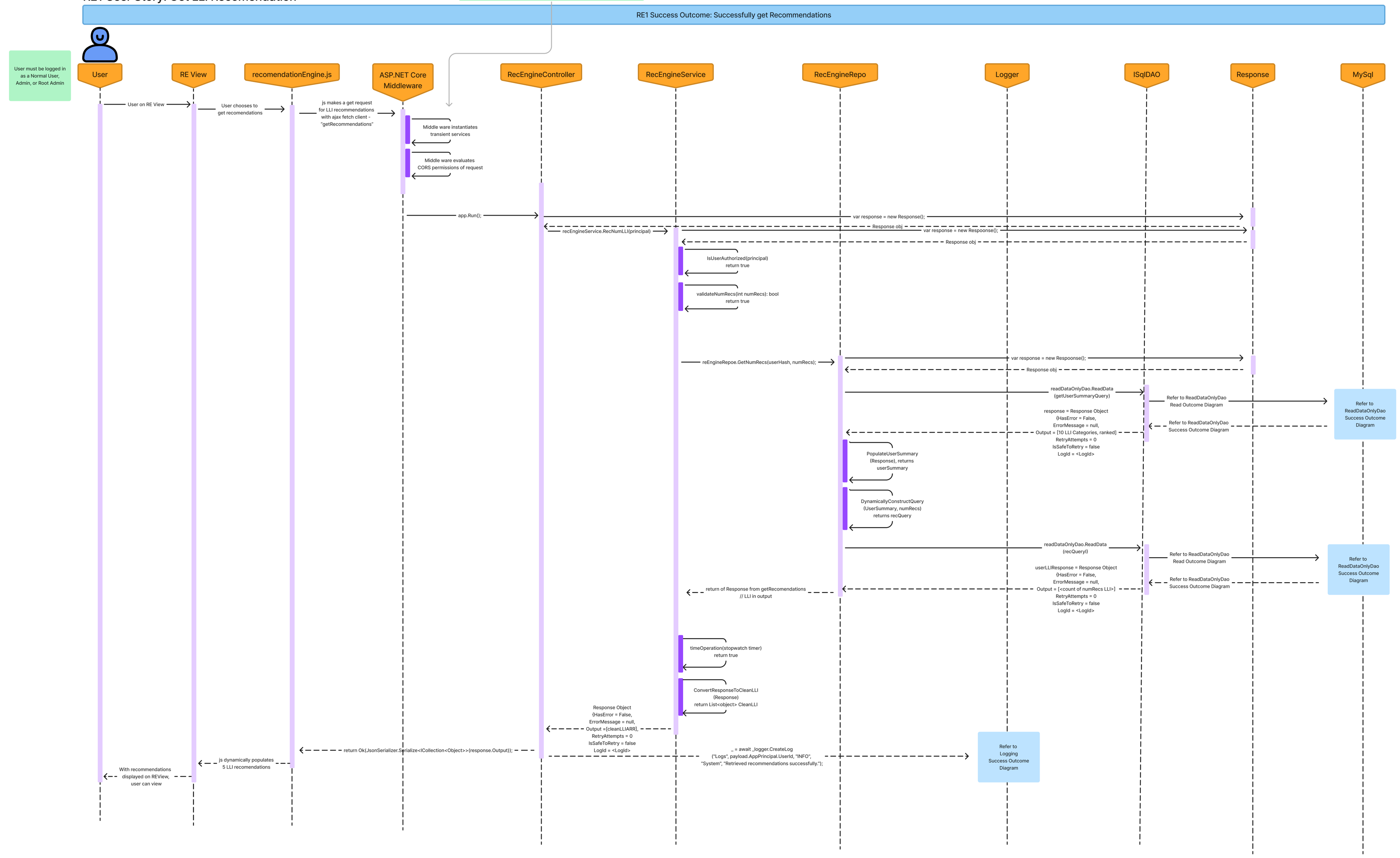
Goals : Extensibility, Security, Performance

Recommendation Engine

RE1 Success Outcome 1

RE1 User Story: Get LLI Recomendation

```
builder.Services.AddTransient<IReadDataOnlyDAO, ReadDataOnlyDAO>();
builder.Services.AddTransient<ICreateDataOnlyDAO, CreateDataOnlyDAO>();
builder.Services.AddTransient<ILogTarget, LogTarget>();
builder.Services.AddTransient<ILogging, Logging>();
builder.Services.AddTransient<IRecEngineRepo, RecEngineRepo>();
builder.Services.AddTransient<IRecEngineService, RecEngineService>();
builder.Services.AddTransient<IJWTService, JWTService>();
builder.Services.AddTransient<ILifelogAuthService, LifelogAuthService>();
```

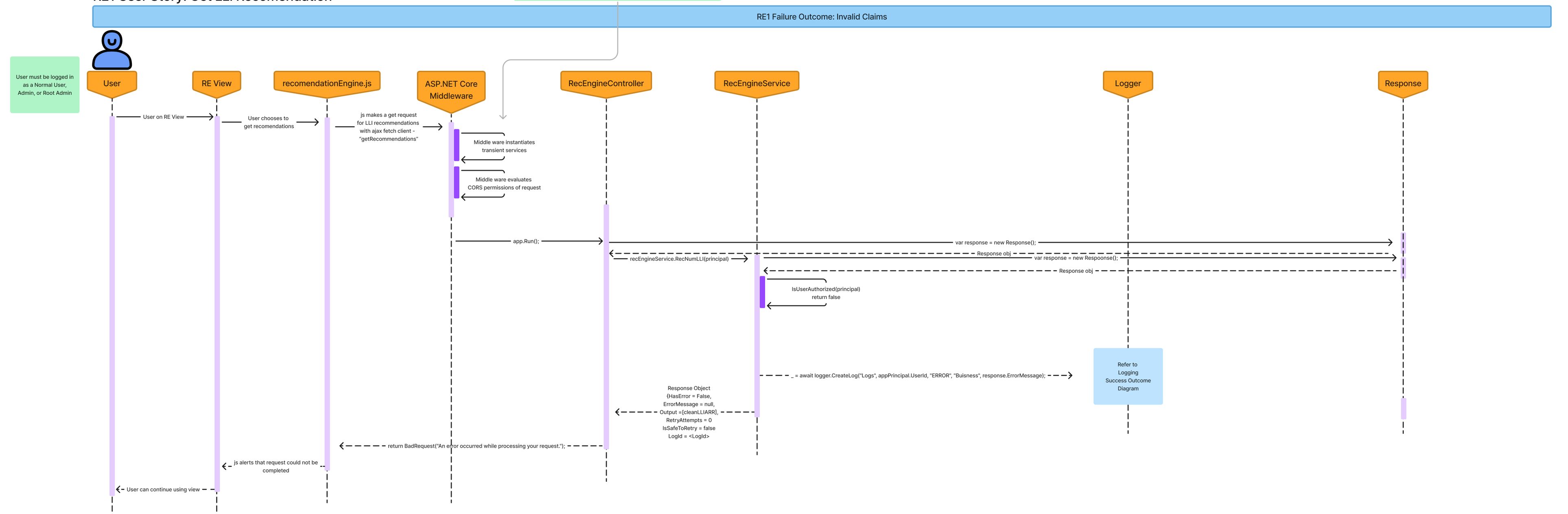

RE1 Success Outcome: Successfully get Recommendations
User must be logged in as a Normal User, Admin, or Root Admin
User
RE View
recomendationEngine.js
ASP.NET Core Middleware
RecEngineController
RecEngineService
RecEngineRepo
Logger
ISqlDAO
Response
MySql
User on RE View
User chooses to get recomendations
js makes a get request for LLI recommendations with ajax fetch client - "getRecommendations"
Middle ware instantiates transient services
Middle ware evaluates CORS permissions of request
app.Run();
var response = new Response();
Response obj
recEngineService.RecNumLLI(principal)
var response = new Respoonse();
Response obj
IsUserAuthorized(principal) return true
validateNumRecs(int numRecs): bool return true
reEngineRepoe.GetNumRecs(userHash, numRecs);
var response = new Respoonse();
Response obj
readDataOnlyDao.ReadData (getUserSummaryQuery)
Refer to ReadDataOnlyDao Read Outcome Diagram
Refer to ReadDataOnlyDao Success Outcome Diagram
Refer to ReadDataOnlyDao Success Outcome Diagram
response = Response Object {HasError = False, ErrorMessage = null, Output = [10 LLI Categories, ranked] RetryAttempts = 0 IsSafeToRetry = false LogId = <LogId>
PopulateUserSummary (Response), returns userSummary
DynamicallyConstructQuery (UserSummary, numRecs) returns recQuery
readDataOnlyDao.ReadData (recQuery)
Refer to ReadDataOnlyDao Read Outcome Diagram
Refer to ReadDataOnlyDao Success Outcome Diagram
Refer to ReadDataOnlyDao Success Outcome Diagram
userLLIResponse = Response Object {HasError = False, ErrorMessage = null, Output = [<count of numRecs LLI>] RetryAttempts = 0 IsSafeToRetry = false LogId = <LogId>
return of Response from getRecomendations // LLI in output
timeOperation(stopwatch timer) return true
ConvertResponseToCleanLLI (Response) return List<object> CleanLLI
Response Object {HasError = False, ErrorMessage = null, Output =[cleanLLIARR], RetryAttempts = 0 IsSafeToRetry = false LogId = <LogId>
return Ok(JsonSerializer.Serialize<ICollection<Object>>(response.Output));
_ = await _logger.CreateLog ("Logs", payload.AppPrincipal.UserId, "INFO", "System", "Retrieved recommendations successfully.");
Refer to Logging Success Outcome Diagram
js dynamically populates 5 LLI recomendations
With recommendations displayed on REView, user can view

Recommendation Engine

RE1 Failure Outcome 1

RE1 User Story: Get LLI Recomendation

```
builder.Services.AddTransient<IReadDataOnlyDAO, ReadDataOnlyDAO>();
builder.Services.AddTransient<ICreateDataOnlyDAO, CreateDataOnlyDAO>();
builder.Services.AddTransient<ILogTarget, LogTarget>();
builder.Services.AddTransient<ILogging, Logging>();
builder.Services.AddTransient<IRecEngineRepo, RecEngineRepo>();
builder.Services.AddTransient<IRecEngineService, RecEngineService>();
builder.Services.AddTransient<IJWTService, JWTService>();
builder.Services.AddTransient<ILifelogAuthService, LifelogAuthService>();
```

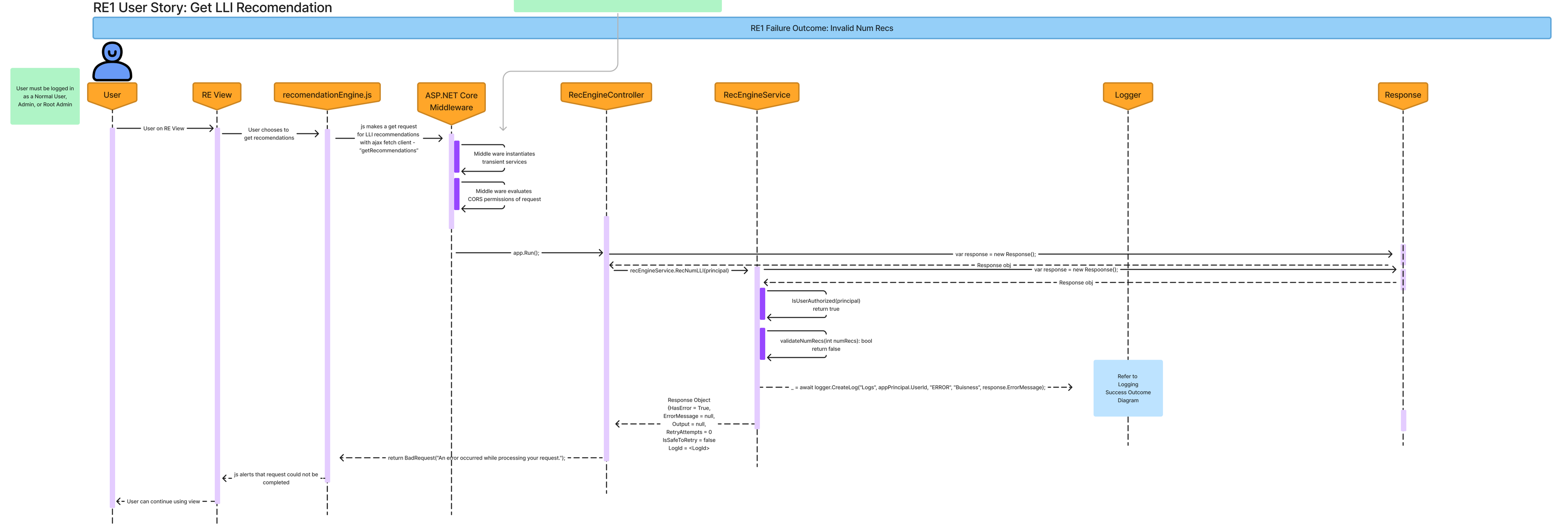

RE1 Failure Outcome: Invalid Claims
User must be logged in as a Normal User, Admin, or Root Admin
User
RE View
recomendationEngine.js
ASP.NET Core Middleware
RecEngineController
RecEngineService
Logger
Response
User on RE View
User chooses to get recomendations
js makes a get request for LLI recommendations with ajax fetch client - "getRecommendations"
Middle ware instantiates transient services
Middle ware evaluates CORS permissions of request
app.Run();
var response = new Response();
Response obj
recEngineService.RecNumLLI(principal)
var response = new Respoonse();
Response obj
IsUserAuthorized(principal) return false
_ = await logger.CreateLog("Logs", appPrincipal.UserId, "ERROR", "Buisness", response.ErrorMessage);
Refer to Logging Success Outcome Diagram
Response Object {HasError = False, ErrorMessage = null, Output =[cleanLLIARR], RetryAttempts = 0 IsSafeToRetry = false LogId = <LogId>
return BadRequest("An error occurred while processing your request.");
js alerts that request could not be completed
User can continue using view

Recommendation Engine

RE1 Failure Outcome 2

RE1 User Story: Get LLI Recomendation

```
builder.Services.AddTransient<IReadDataOnlyDAO, ReadDataOnlyDAO>();
builder.Services.AddTransient<ICreateDataOnlyDAO, CreateDataOnlyDAO>();
builder.Services.AddTransient<ILogTarget, LogTarget>();
builder.Services.AddTransient<ILogging, Logging>();
builder.Services.AddTransient<IRecEngineRepo, RecEngineRepo>();
builder.Services.AddTransient<IRecEngineService, RecEngineService>();
builder.Services.AddTransient<IJWTService, JWTService>();
builder.Services.AddTransient<ILifelogAuthService, LifelogAuthService>();
```

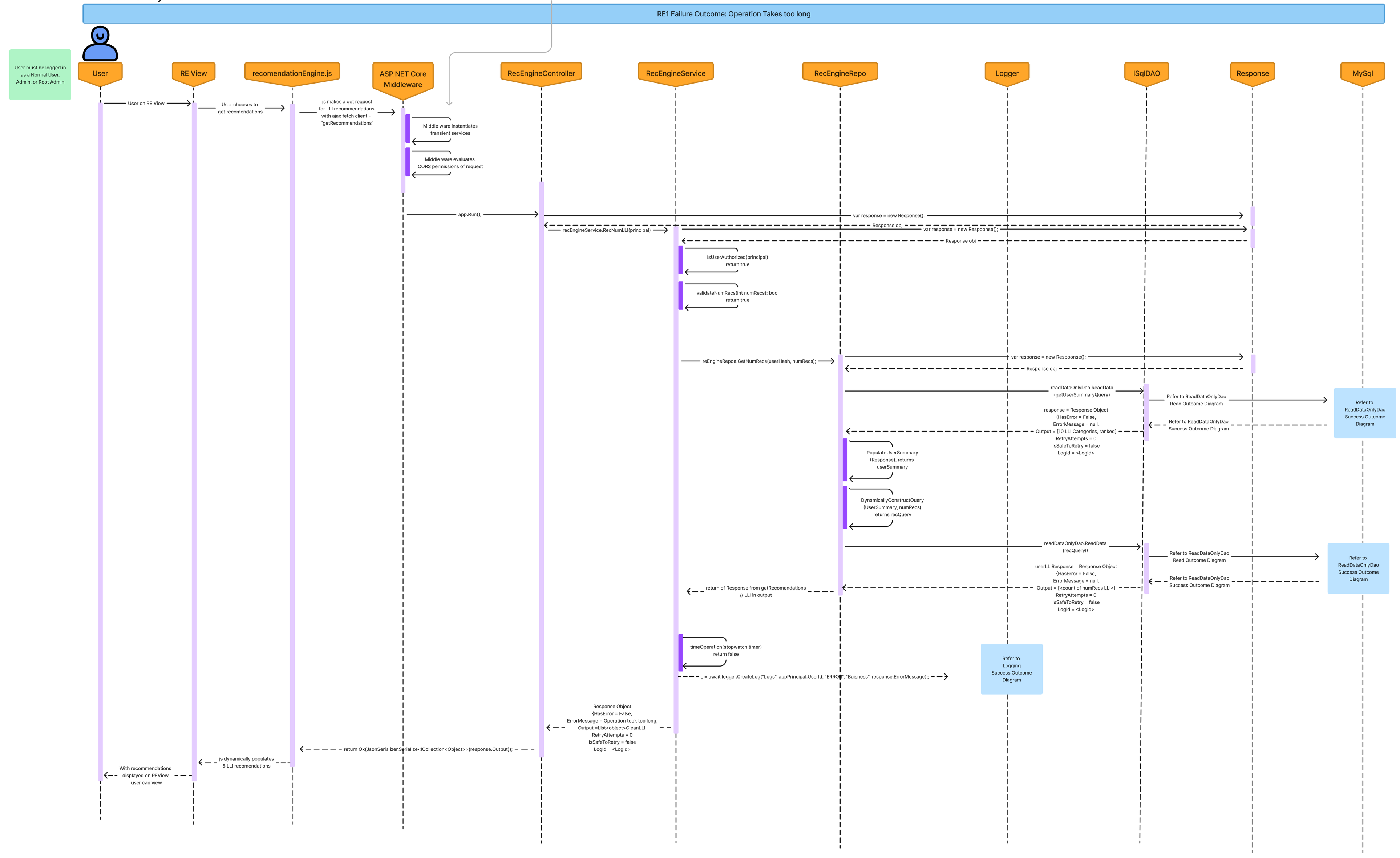

RE1 Failure Outcome: Invalid Num Recs
User must be logged in as a Normal User, Admin, or Root Admin
User
RE View
recomendationEngine.js
ASP.NET Core Middleware
RecEngineController
RecEngineService
Logger
Response
User on RE View
User chooses to get recomendations
js makes a get request for LLI recommendations with ajax fetch client - "getRecommendations"
Middle ware instantiates transient services
Middle ware evaluates CORS permissions of request
app.Run();
var response = new Response();
Response obj
recEngineService.RecNumLLI(principal)
var response = new Respoonse();
Response obj
IsUserAuthorized(principal) return true
validateNumRecs(int numRecs): bool return false
_ = await logger.CreateLog("Logs", appPrincipal.UserId, "ERROR", "Buisness", response.ErrorMessage);
Refer to Logging Success Outcome Diagram
Response Object {HasError = True, ErrorMessage = null, Output = null, RetryAttempts = 0 IsSafeToRetry = false LogId = <LogId>
return BadRequest("An error occurred while processing your request.");
js alerts that request could not be completed
User can continue using view

Recommendation Engine

RE1 Failure Outcome 3

RE1 User Story: Get LLI Recomendation

```
builder.Services.AddTransient<IReadDataOnlyDAO, ReadDataOnlyDAO>();
builder.Services.AddTransient<ICreateDataOnlyDAO, CreateDataOnlyDAO>();
builder.Services.AddTransient<ILogTarget, LogTarget>();
builder.Services.AddTransient<ILogging, Logging>();
builder.Services.AddTransient<IRecEngineRepo, RecEngineRepo>();
builder.Services.AddTransient<IRecEngineService, RecEngineService>();
builder.Services.AddTransient<IJWTService, JWTService>();
builder.Services.AddTransient<ILifelogAuthService, LifelogAuthService>();
```

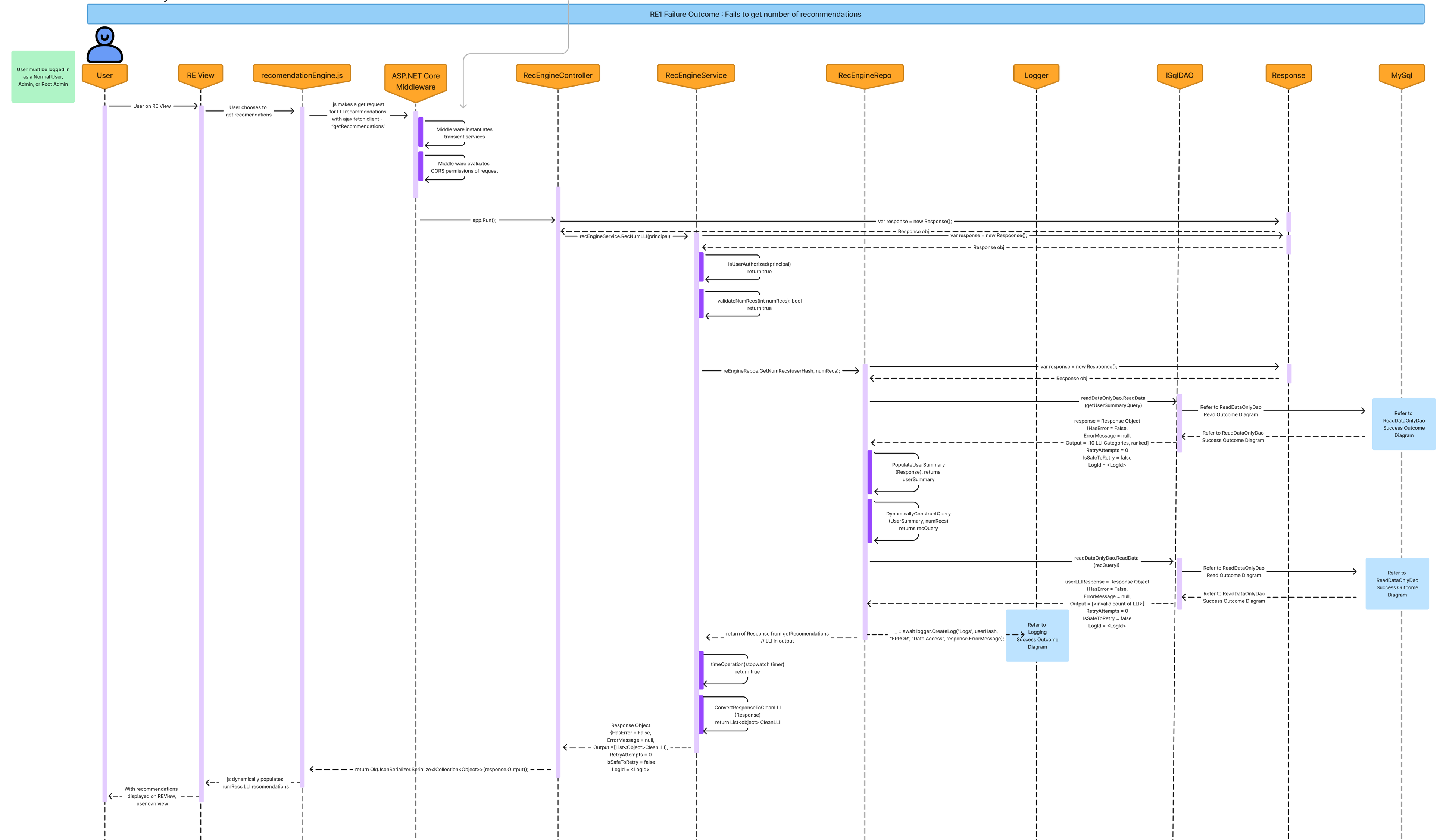

RE1 Failure Outcome: Operation Takes too long
User must be logged in as a Normal User, Admin, or Root Admin
User
RE View
recomendationEngine.js
ASP.NET Core Middleware
RecEngineController
RecEngineService
RecEngineRepo
Logger
ISqlDAO
Response
MySql
User on RE View
User chooses to get recomendations
js makes a get request for LLI recommendations with ajax fetch client - "getRecommendations"
Middle ware instantiates transient services
Middle ware evaluates CORS permissions of request
app.Run();
var response = new Response();
Response obj
recEngineService.RecNumLLI(principal)
var response = new Respoonse();
Response obj
IsUserAuthorized(principal) return true
validateNumRecs(int numRecs): bool return true
reEngineRepoe.GetNumRecs(userHash, numRecs);
var response = new Respoonse();
Response obj
readDataOnlyDao.ReadData (getUserSummaryQuery)
Refer to ReadDataOnlyDao Read Outcome Diagram
Refer to ReadDataOnlyDao Success Outcome Diagram
Refer to ReadDataOnlyDao Success Outcome Diagram
response = Response Object {HasError = False, ErrorMessage = null, Output = [10 LLI Categories, ranked] RetryAttempts = 0 IsSafeToRetry = false LogId = <LogId>
PopulateUserSummary (Response), returns userSummary
DynamicallyConstructQuery (UserSummary, numRecs) returns recQuery
readDataOnlyDao.ReadData (recQuery)
Refer to ReadDataOnlyDao Read Outcome Diagram
Refer to ReadDataOnlyDao Success Outcome Diagram
Refer to ReadDataOnlyDao Success Outcome Diagram
userLLIResponse = Response Object {HasError = False, ErrorMessage = null, Output = [<count of numRecs LLI>] RetryAttempts = 0 IsSafeToRetry = false LogId = <LogId>
return of Response from getRecomendations // LLI in output
timeOperation(stopwatch timer) return false
_ = await logger.CreateLog("Logs", appPrincipal.UserId, "ERROR", "Buisness", response.ErrorMessage);
Refer to Logging Success Outcome Diagram
Response Object {HasError = False, ErrorMessage = Operation took too long, Output =<List<object>CleanLLI>, RetryAttempts = 0 IsSafeToRetry = false LogId = <LogId>
return Ok(JsonSerializer.Serialize<ICollection<Object>>(response.Output));
js dynamically populates 5 LLI recomendations
With recommendations displayed on REView, user can view

Recommendation Engine

RE1 Failure Outcome 4

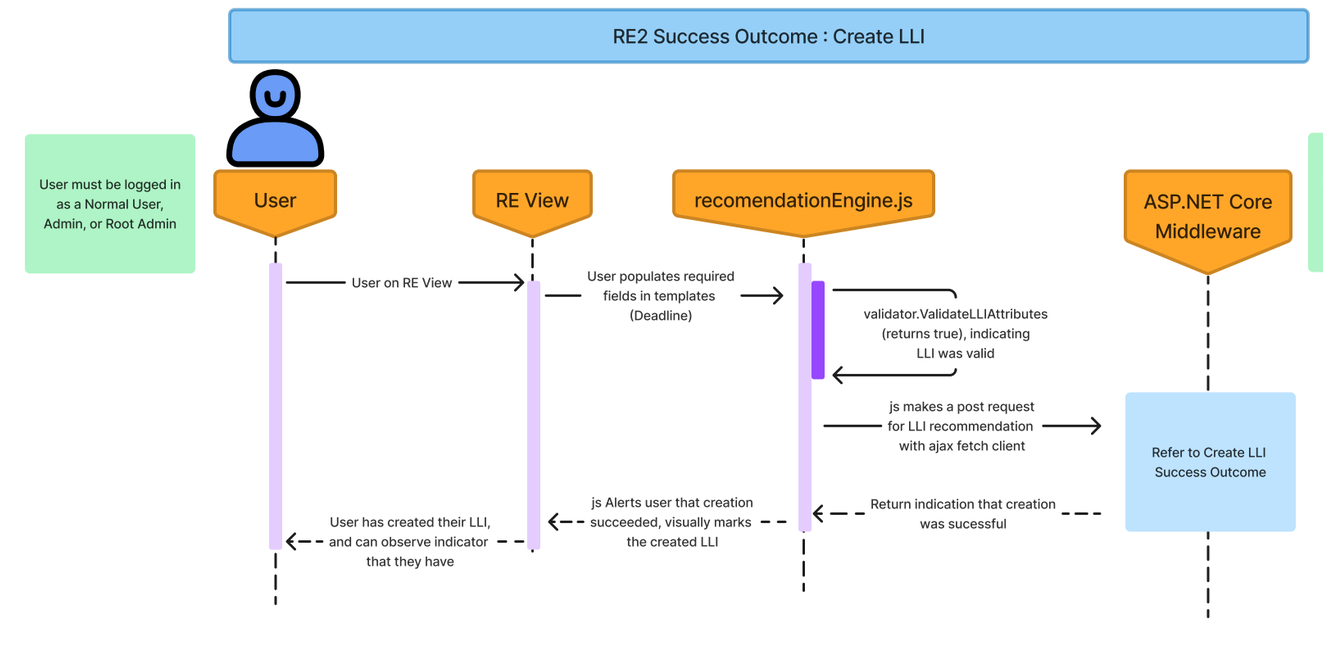
RE1 User Story: Get LLI Recomendation

```
builder.Services.AddTransient<IReadDataOnlyDAO, ReadDataOnlyDAO>();
builder.Services.AddTransient<ICreateDataOnlyDAO, CreateDataOnlyDAO>();
builder.Services.AddTransient<ILogTarget, LogTarget>();
builder.Services.AddTransient<ILogging, Logging>();
builder.Services.AddTransient<IRecEngineRepo, RecEngineRepo>();
builder.Services.AddTransient<IRecEngineService, RecEngineService>();
builder.Services.AddTransient<IJWTService, JWTService>();
builder.Services.AddTransient<ILifelogAuthService, LifelogAuthService>();
```


RE1 Failure Outcome : Fails to get number of recommendations
User must be logged in as a Normal User, Admin, or Root Admin
User
RE View
recomendationEngine.js
ASP.NET Core Middleware
RecEngineController
RecEngineService
RecEngineRepo
Logger
ISqlDAO
Response
MySql
User on RE View
User chooses to get recomendations
js makes a get request for LLI recommendations with ajax fetch client - "getRecommendations"
Middle ware instantiates transient services
Middle ware evaluates CORS permissions of request
app.Run();
var response = new Response();
Response obj
recEngineService.RecNumLLI(principal)
var response = new Respoonse();
Response obj
IsUserAuthorized(principal) return true
validateNumRecs(int numRecs): bool return true
reEngineRepoe.GetNumRecs(userHash, numRecs);
var response = new Respoonse();
Response obj
readDataOnlyDao.ReadData (getUserSummaryQuery)
Refer to ReadDataOnlyDao Read Outcome Diagram
Refer to ReadDataOnlyDao Success Outcome Diagram
Refer to ReadDataOnlyDao Success Outcome Diagram
response = Response Object {HasError = False, ErrorMessage = null, Output = [10 LLI Categories, ranked] RetryAttempts = 0 IsSafeToRetry = false LogId = <LogId>
PopulateUserSummary (Response), returns userSummary
DynamicallyConstructQuery (UserSummary, numRecs) returns recQuery
readDataOnlyDao.ReadData (recQuery)
Refer to ReadDataOnlyDao Read Outcome Diagram
Refer to ReadDataOnlyDao Success Outcome Diagram
Refer to ReadDataOnlyDao Success Outcome Diagram
userLLIResponse = Response Object {HasError = False, ErrorMessage = null, Output = [<invalid count of LLI>] RetryAttempts = 0 IsSafeToRetry = false LogId = <LogId>
_ = await logger.CreateLog("Logs", userHash, "ERROR", "Data Access", response.ErrorMessage);
Refer to Logging Success Outcome Diagram
return of Response from getRecomendations // LLI in output
timeOperation(stopwatch timer) return true
ConvertResponseToCleanLLI (Response) return List<object> CleanLLI
Response Object {HasError = False, ErrorMessage = null, Output =[List<Object>CleanLLI], RetryAttempts = 0 IsSafeToRetry = false LogId = <LogId>
return Ok(JsonSerializer.Serialize<ICollection<Object>>(response.Output));
js dynamically populates numRecs LLI recomendations
With recommendations displayed on REView, user can view

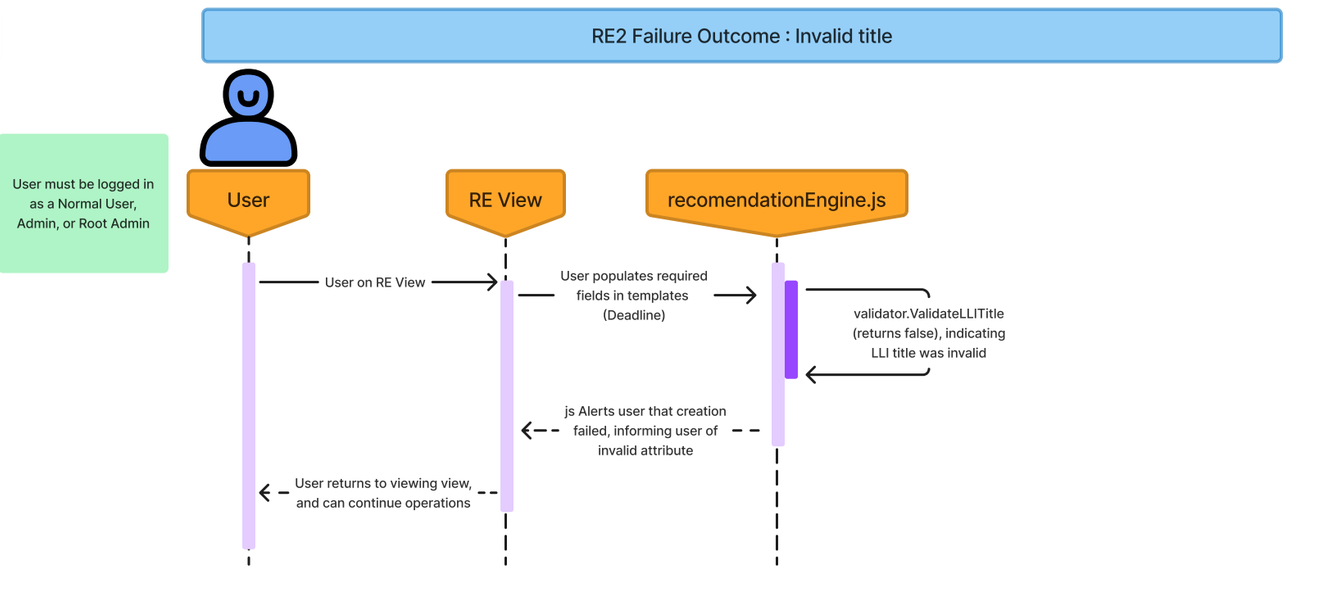
RE2 Success Outcome 1

RE2 User Story: Create Recommended LLI


RE2 Success Outcome : Create LLI
User must be logged in as a Normal User, Admin, or Root Admin
User
RE View
recomendationEngine.js
ASP.NET Core Middleware
User on RE View
User populates required fields in templates (Deadline)
validator.ValidateLLIAttributes (returns true), indicating LLI was valid
js makes a post request for LLI recommendation with ajax fetch client
Refer to Create LLI Success Outcome
Return indication that creation was sucessful
js Alerts user that creation succeeded, visually marks the created LLI
User has created their LLI, and can observe indicator that they have

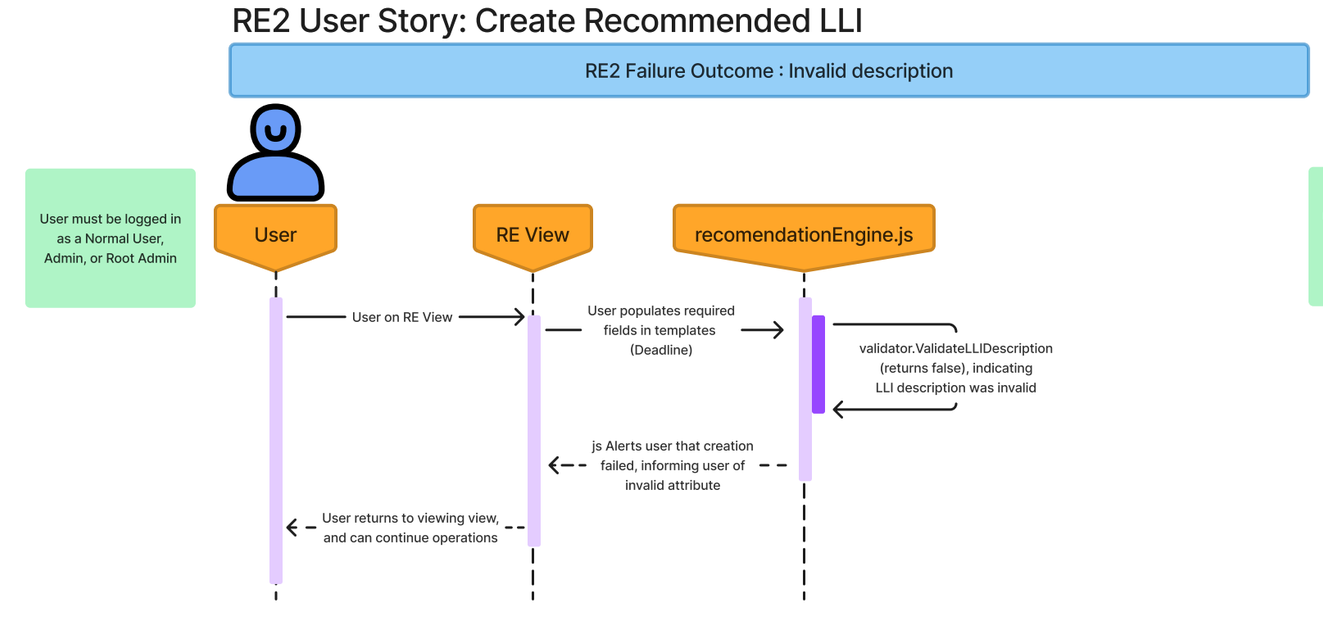
RE2 Failure Outcome 1

RE2 User Story: Create Recommended LLI


RE2 Failure Outcome : Invalid title
User must be logged in as a Normal User, Admin, or Root Admin
User
RE View
recomendationEngine.js
User on RE View
User populates required fields in templates (Deadline)
validator.ValidateLLITitle (returns false), indicating LLI title was invalid
js Alerts user that creation failed, informing user of invalid attribute
User returns to viewing view, and can continue operations

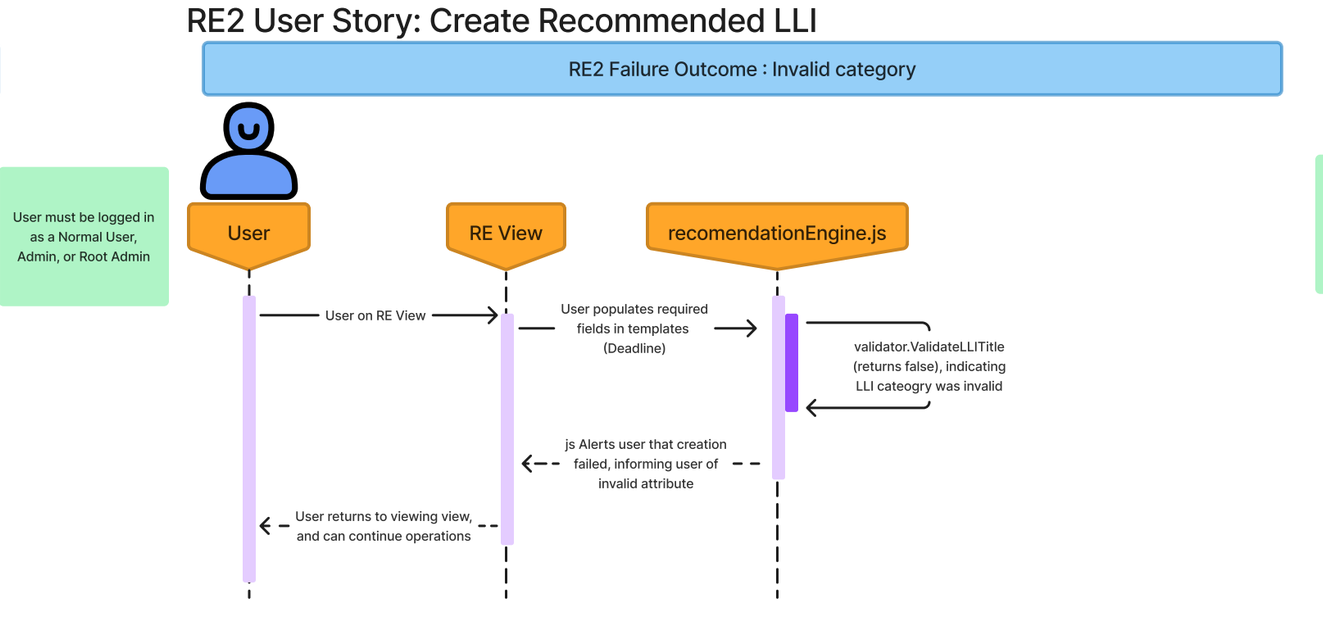
RE2 Failure Outcome 2

RE2 User Story: Create Recommended LLI


RE2 Failure Outcome : Invalid description
User must be logged in as a Normal User, Admin, or Root Admin
User
RE View
recomendationEngine.js
User on RE View
User populates required fields in templates (Deadline)
validator.ValidateLLIDescription (returns false), indicating LLI description was invalid
js Alerts user that creation failed, informing user of invalid attribute
User returns to viewing view, and can continue operations

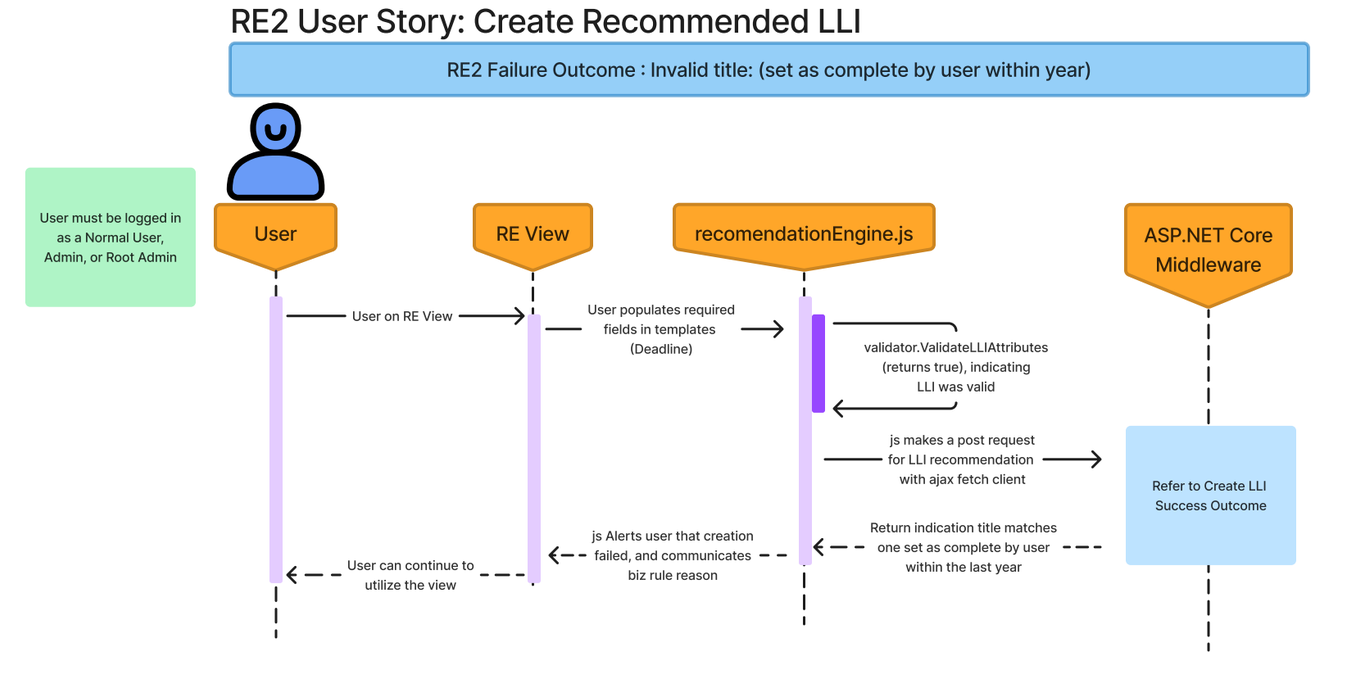
RE2 Failure Outcome 3

RE2 User Story: Create Recommended LLI


RE2 Failure Outcome : Invalid category
User must be logged in as a Normal User, Admin, or Root Admin
User
RE View
recomendationEngine.js
User on RE View
User populates required fields in templates (Deadline)
validator.ValidateLLITitle (returns false), indicating LLI category was invalid
js Alerts user that creation failed, informing user of invalid attribute
User returns to viewing view, and can continue operations

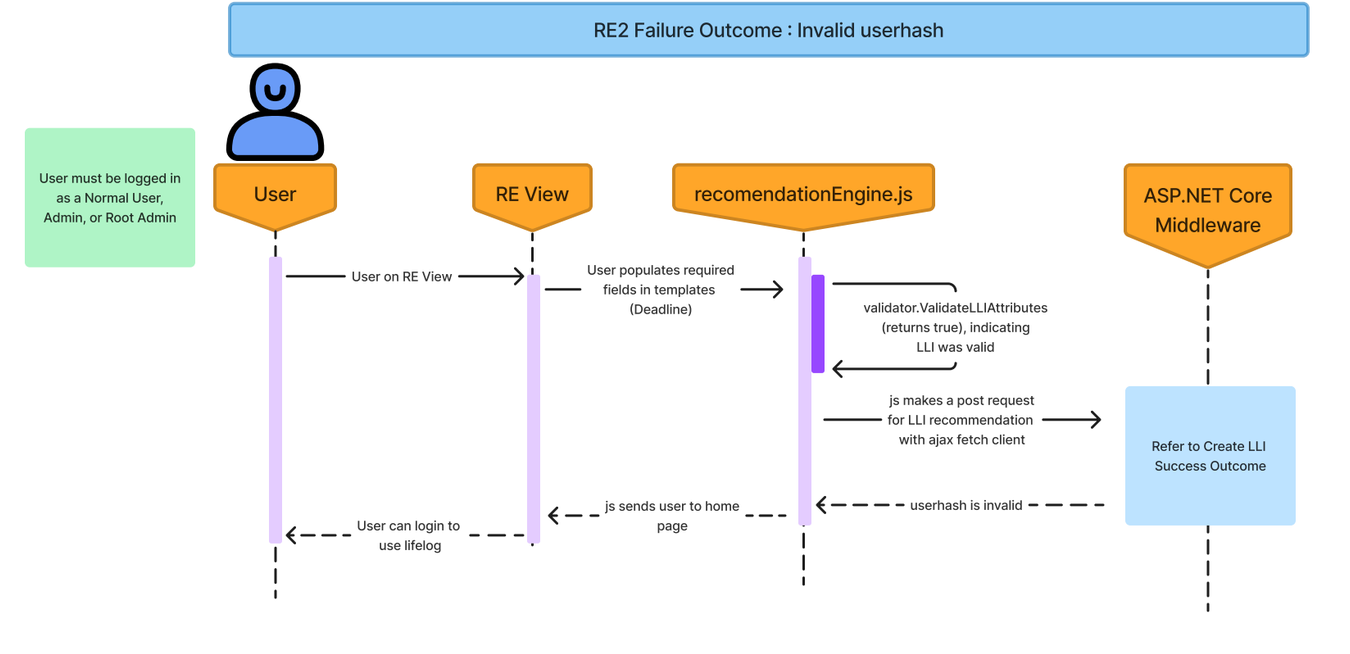
RE2 Failure Outcome 4

RE2 User Story: Create Recommended LLI


RE2 Failure Outcome : Invalid title: (set as complete by user within year)
User must be logged in as a Normal User, Admin, or Root Admin
User
RE View
recomendationEngine.js
ASP.NET Core Middleware
User on RE View
User populates required fields in templates (Deadline)
validator.ValidateLLIAttributes (returns true), indicating LLI was valid
js makes a post request for LLI recommendation with ajax fetch client
Refer to Create LLI Success Outcome
Return indication title matches one set as complete by user within the last year
js Alerts user that creation failed, and communicates biz rule reason
User can continue to utilize the view

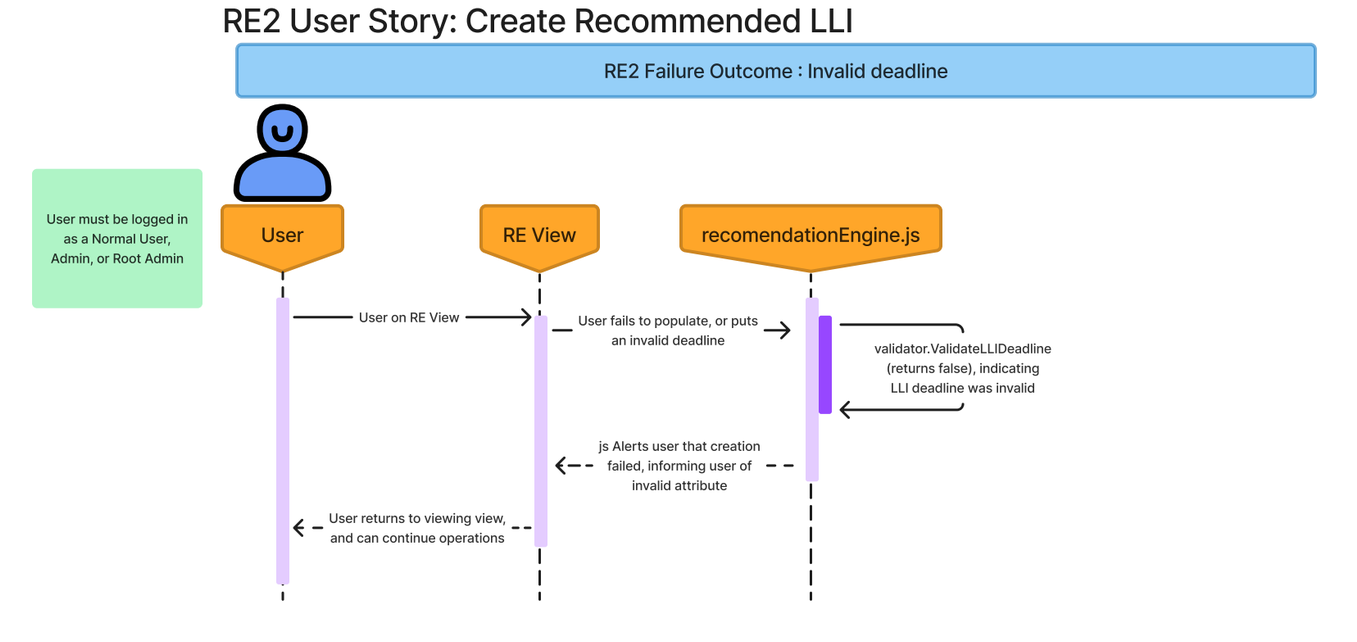
RE2 Failure Outcome 5

RE2 User Story: Create Recommended LLI


RE2 Failure Outcome : Invalid userhash
User must be logged in as a Normal User, Admin, or Root Admin
User
RE View
recomendationEngine.js
ASP.NET Core Middleware
User on RE View
User populates required fields in templates (Deadline)
validator.ValidateLLIAttributes (returns true), indicating LLI was valid
js makes a post request for LLI recommendation with ajax fetch client
Refer to Create LLI Success Outcome
userhash is invalid
js sends user to home page
User can login to use lifelog

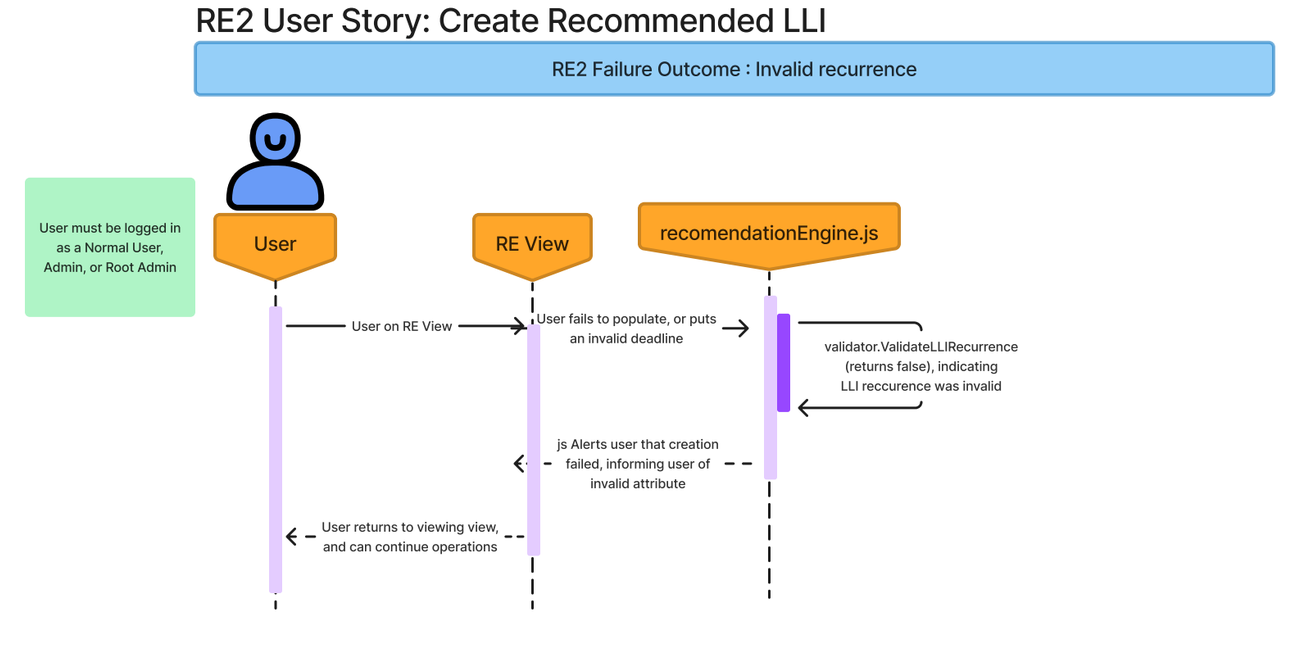
RE2 Failure Outcome 6

RE2 User Story: Create Recommended LLI


RE2 Failure Outcome : Invalid deadline
User must be logged in as a Normal User, Admin, or Root Admin
User
RE View
recomendationEngine.js
User on RE View
User fails to populate, or puts an invalid deadline
validator.ValidateLLIDeadline (returns false), indicating LLI deadline was invalid
js Alerts user that creation failed, informing user of invalid attribute
User returns to viewing view, and can continue operations

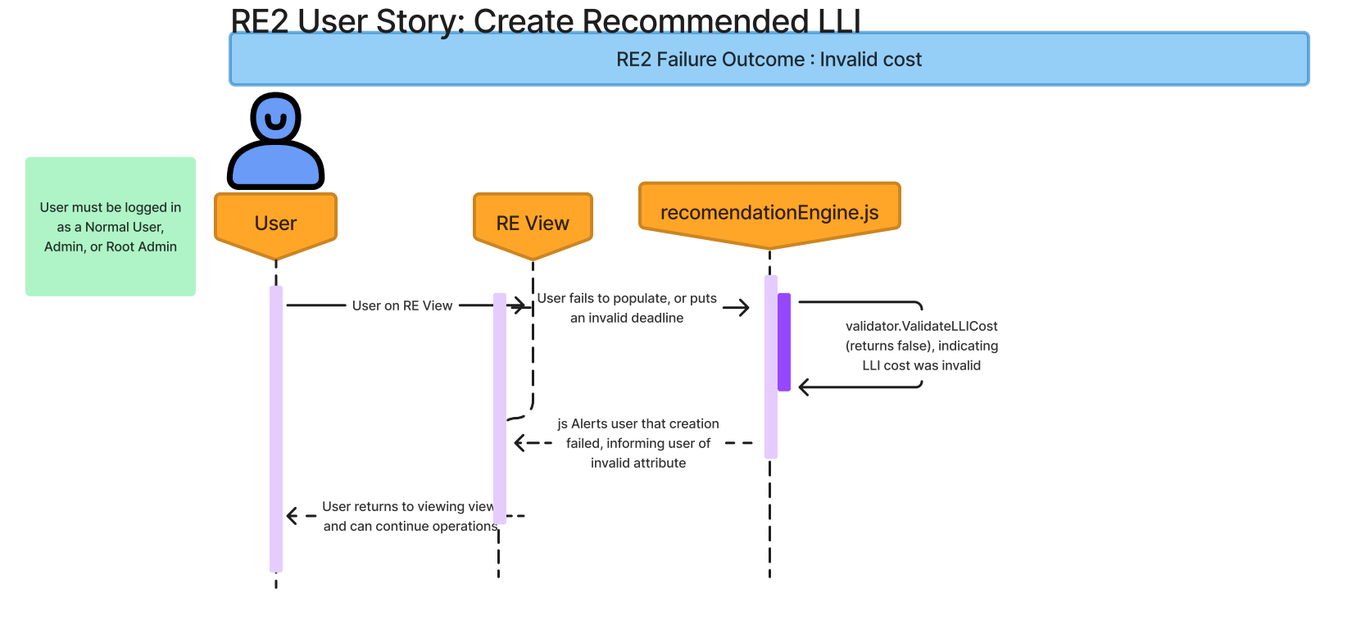
RE2 Failure Outcome 7

RE2 User Story: Create Recommended LLI


RE2 Failure Outcome : Invalid recurrence
User must be logged in as a Normal User, Admin, or Root Admin
User
RE View
recomendationEngine.js
User on RE View
User fails to populate, or puts an invalid deadline
validator.ValidateLLIRecurrence (returns false), indicating LLI recurrence was invalid
js Alerts user that creation failed, informing user of invalid attribute
User returns to viewing view, and can continue operations

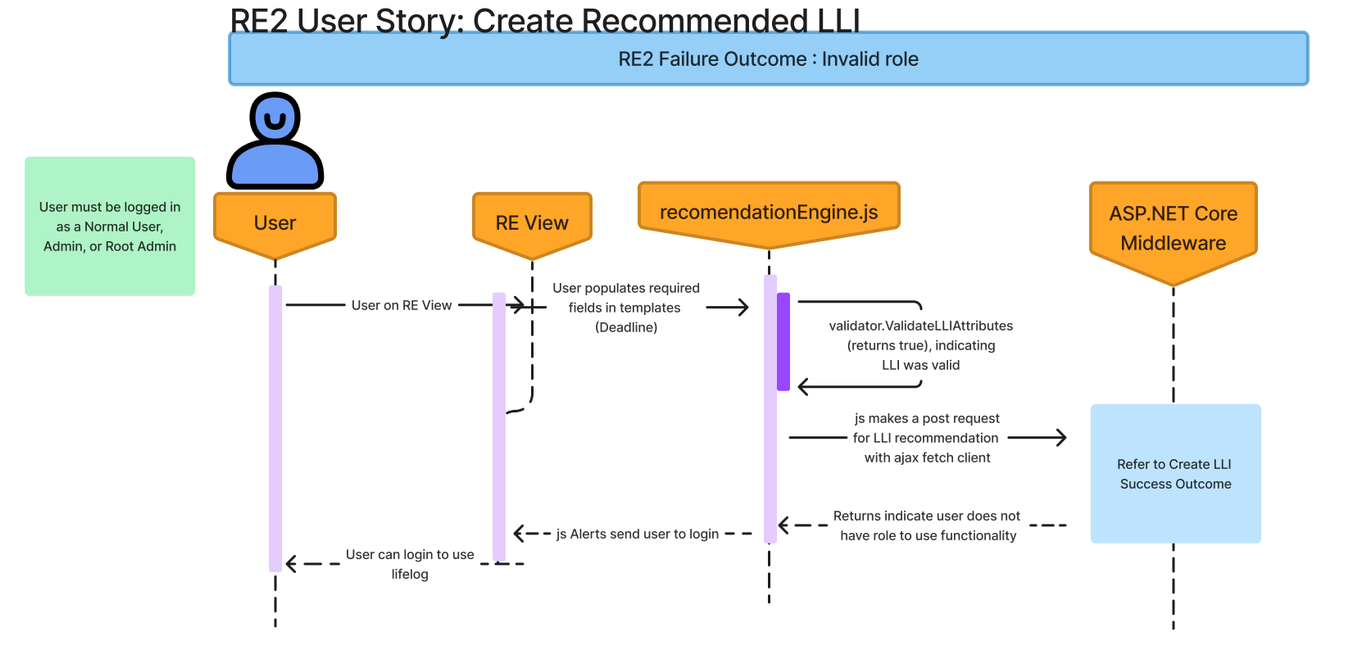
RE2 Failure Outcome 8

RE2 User Story: Create Recommended LLI


RE2 Failure Outcome : Invalid cost
User must be logged in as a Normal User, Admin, or Root Admin
User
RE View
recomendationEngine.js
User on RE View
User fails to populate, or puts an invalid deadline
validator.ValidateLLICost (returns false), indicating LLI cost was invalid
js Alerts user that creation failed, informing user of invalid attribute
User returns to viewing view, and can continue operations

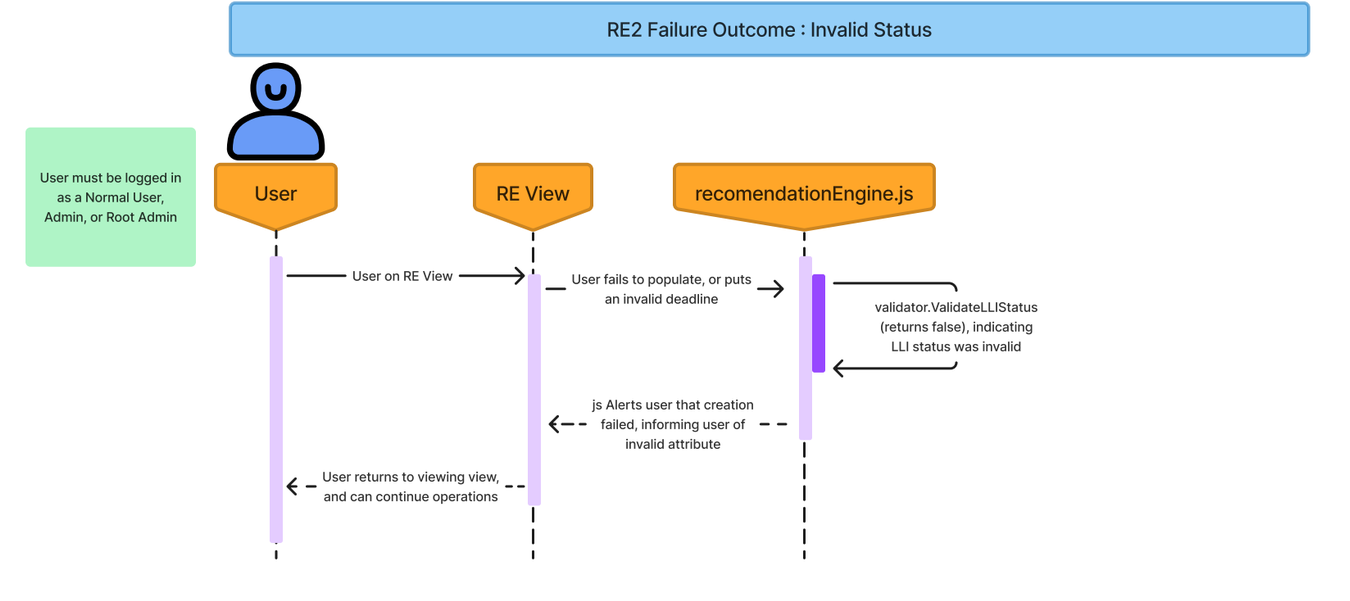
RE2 Failure Outcome 9

RE2 User Story: Create Recommended LLI


RE2 Failure Outcome : Invalid role
User must be logged in as a Normal User, Admin, or Root Admin
User
RE View
recomendationEngine.js
ASP.NET Core Middleware
User on RE View
User populates required fields in templates (Deadline)
validator.ValidateLLIAttributes (returns true), indicating LLI was valid
js makes a post request for LLI recommendation with ajax fetch client
Refer to Create LLI Success Outcome
Returns indicate user does not have role to use functionality
js Alerts send user to login
User can login to use lifelog

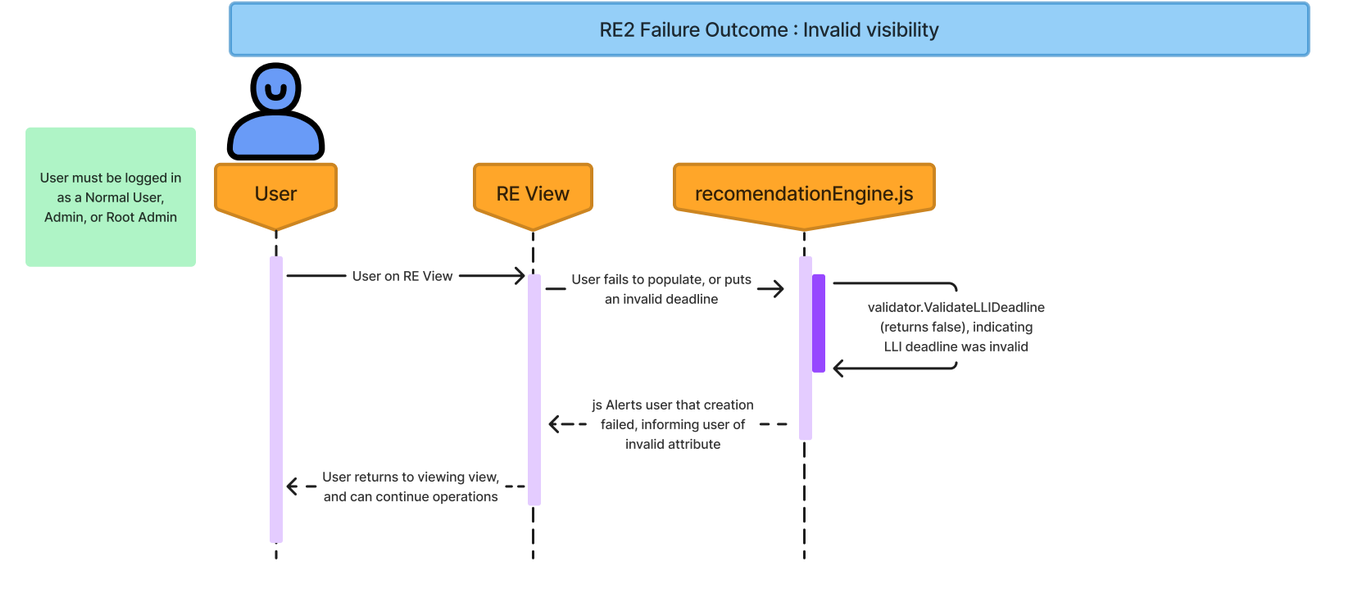
RE2 Failure Outcome 10

RE2 User Story: Create Recommended LLI


RE2 Failure Outcome : Invalid Status
User must be logged in as a Normal User, Admin, or Root Admin
User
RE View
recomendationEngine.js
User on RE View
User fails to populate, or puts an invalid deadline
validator.ValidateLLIStatus (returns false), indicating LLI status was invalid
js Alerts user that creation failed, informing user of invalid attribute
User returns to viewing view, and can continue operations

RE2 Failure Outcome 11

RE2 User Story: Create Recommended LLI


RE2 Failure Outcome : Invalid visibility
User must be logged in as a Normal User, Admin, or Root Admin
User
RE View
recomendationEngine.js
User on RE View
User fails to populate, or puts an invalid deadline
validator.ValidateLLIDeadline (returns false), indicating LLI deadline was invalid
js Alerts user that creation failed, informing user of invalid attribute
User returns to viewing view, and can continue operations