|  |  |
| --- | --- |
|  | **Cognizant Academy**  **Employee Travel Desk**  **FSE – Business Aligned Project**  **Case Study Specification**  **Version 1.0** |
| |  |  |  |  | | --- | --- | --- | --- | |  | **Prepared By / Last Updated By** | **Reviewed By** | **Approved By** | | **Name** | Khaleelullah Hussaini Syed |  |  | | **Role** | Trainer |  |  | | **Signature** | t-syed8 |  |  | | **Date** | 30 August 2022 |  |  | |
|  |

Table of Contents

[1.0 Important Instructions 3](#_Toc117007554)

[2.0 Introduction 4](#_Toc117007555)

[2.1 Purpose of this document 4](#_Toc117007556)

[2.2 Project Overview 4](#_Toc117007557)

[2.3 Scope 4](#_Toc117007558)

[3.0 Use Case Diagram 6](#_Toc117007559)

[4.0 System Architecture Diagram 7](#_Toc117007560)

[5.0 Development Phases 7](#_Toc117007561)

[6.0 System Requirements 8](#_Toc117007562)

[**6.1.1** **Module – Human resources** 8](#_Toc117007563)

[**6.1.2** **Module – Travel Planner** 11](#_Toc117007564)

[**6.1.3** **Module – Reservations management** 15](#_Toc117007565)

[**6.1.4** **Module – Reimbursement management** 19](#_Toc117007566)

[7.0 Deployment requirements 22](#_Toc117007567)

[8.0 Design Considerations 22](#_Toc117007568)

[8.1.1 Most Important and Common rules 23](#_Toc117007569)

[9.0 Reference learning 24](#_Toc117007570)

[10. Project Templates 25](#_Toc117007571)

[11. Change Log 25](#_Toc117007572)

# Important Instructions

1. Associate must adhere to the Design Considerations specific to each Technolgy Track.
2. Associate must not submit project with compile-time or build-time errors.
3. Being a Full-Stack Developer Project, you must focus on ALL layers of the application development.
4. Unit Testing is Mandatory, and we expect a code coverage of 100%. Use Unit testing and Mocking Frameworks wherever applicable.
5. All the Microservices, Client Application, DB Scripts, have to be packaged together in a single ZIP file. Associate must submit the solution file in ZIP format only.
6. If back end has to be set up manually, appropriate DB scripts have to be provided along with the solution ZIP file.
7. A READ ME has to be provided with steps to execute the submitted solution, the Launch URLs of the Microservices in cloud must be specified.

(Importantly, the READ ME should contain the steps to execute DB scripts, the LAUNCH URL of the application)

1. Follow coding best practices while implementing the solution. Use appropriate design patterns wherever applicable.
2. You are supposed to use an In-memory database or code level data as specified, for the Microservices that should be deployed in cloud. No Physical database is suggested for Microservice.

# Introduction

## Purpose of this document

The purpose of the software requirement document is to systematically capture requirements for the project and the system “Employee Travel Desk” that has to be developed. Both functional and non-functional requirements are captured in this document. It also serves as the input for the project scoping.

The scope of this document is limited to addressing the requirements from a user, quality, and non-functional perspective.

High Level Design considerations are also specificed wherever applicable, however the detailed design considerations have to be strictly adhered to during implementation.

## Project Overview

Employees of the cognizant technology solutions are often required to visit the various client locations for the business purposes. The Employee Travel Desk Portal will now provided an automated workflow for planning the travel of employees with due approvals from their HR. The portal further will provide the facilities for doing reservations tickets and reimbursements of expenses incurred during the stay.

## Scope

Below are the modules that needs to be developed part of the Project:

|  |  |  |
| --- | --- | --- |
| **Req. No.** | **Req. Name** | **Req. Description** |
| REQ\_01 | **Human Resource** module | * This module will be used by HRs to manage the employee records with-in the organization. * The HR will be able to add a new employee into the system * The HR can delete records of an existing employee * The HR can also update the current grade of the existing employee in the system |
| REQ\_02 | **Travel Planner** module | * The travel planner module will allow the employees who have access to the system to raise the request for the travel * This module will allow a HR to approve or reject a travel request raised by an employee * An employee can also track the status of their travel request |
| REQ\_03 | **Reservations** module | * In order to travel the employees will requires various reservations like transport, hotel and cab. This module will allow the travel desk execute to upload the reservation details into the system * An employee will be able to track the reservation status for his/her travels * Once the reservation details are available an employee will also be able to download the required documents |
| REQ\_04 | **Reimbursements** module | * An employee will incur various petty expenses on a daily basis during their stay at the travel location. * This module is will help the employees in applying for reimbursements * The helpdesk executives can verify and process the reimbursements * Whereas the HR can see the total expenses incurred during the stay as a report. |

Table 1 : Application Modules

# Use Case Diagram

The following use case diagram shows various users of the system and their responsibilities.



Figure 1 : Use case diagram

# System Architecture Diagram

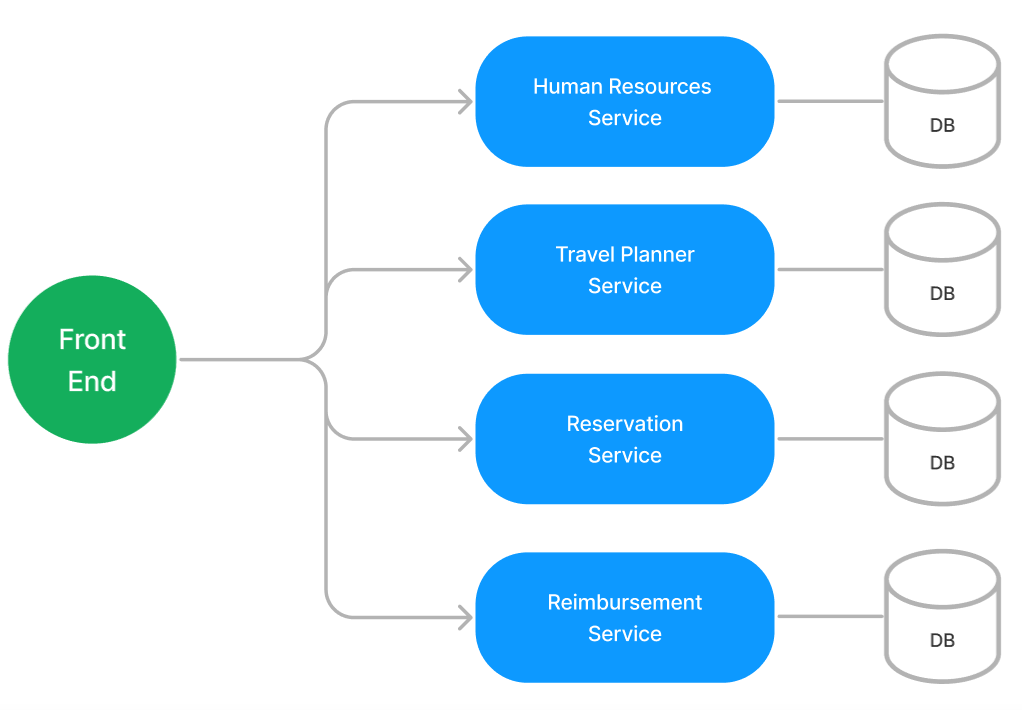


Figure 2 : Application Architecture Diagram

# Development Phases

* The application will be developed in 2 phase.
* Each phase will have 4 stages followed by a review at the end.
* The phase-1 output will be unit tested core business logic of the application.
* In phase-2 the output will be a functional application with micro-service and the font-end.
* Each stage of the development phase must be completed alongside the learning milestone

# System Requirements

### **Module – Human resources**

The account management module will provide the cognizant HR with the following features

1. Add an employee into the system
2. Remove an existing employee
3. Change the current employee grade

**Stage: Database Implementation**

1. Design a data base as per the following ER diagram provided.

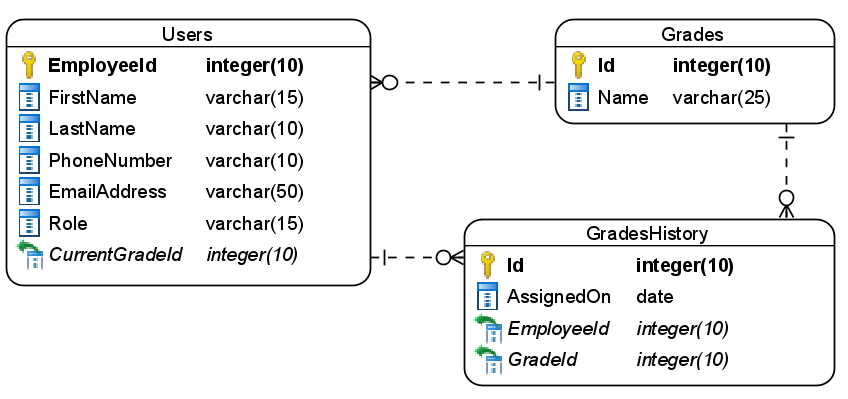


Figure 3 : ER Diagram – Account management

1. Enforce the following constraints on the database apart from primary key, foreign key and unique keys
   1. Phone number must be exactly 10 characters
   2. The default value for access granted must be true
   3. The values allowed for the role are – Employee, HR, TravelDeskExe

**Note**: Pre-populate an HR and TravelDeskExec into the database on the application startup. Also seed few grades into the grades table

**Stage: Data Access Layer Design**

1. Create a library project and add ORM support into it.
2. Use the ORM to map the entities to database as per the ER diagram provided.
3. Use repository per entity pattern and generate the repositories to perform the following operations
   1. Insert a new employees
   2. Return employees list
   3. Delete an employee record
   4. Update the employee grade
   5. Return an employee by ID
   6. Return grades list

**Stage: Business Logic Layer Development**

1. Develop a library which reference the Data Access Library project created earlier
2. This class library will contain various service classes which will encapsulate the business logic for the application.
3. Use dependency injection to in service classes to inject the required repositories.
4. Create the service classes following the single responsibility principle which perform the given operations as follows
   1. Return a list of employees
   2. Add a new employee
   3. Delete an employee
   4. Update an employee
   5. Find an employee by id
   6. Return grades list
5. Following business rules must be implemented as part of the business service class
   1. An employee ID must be a valid 6 digit number
   2. Email address must always be [xxxx@cognizant.com](mailto:xxxx@cognizant.com) where xxxx can be any other value.
   3. Grade of an employee should be allowed to go upwards not downwards, for example a Grade-1 employee cannot be downgraded to Grade-2.
   4. An employee’s grade can only be changed once in an year. The grade of new joinees can only be changed after they complete 2 years. If a user tries to break the rule for grading generate a user-defined exception as “GradeUpdateRuleViolationException”.
   5. When a new TravelDeskExec is added by HR the default grade should be Grade-1

**Stage: Unit Testing**

1. Create a new Unit test project to test the service classes created in business logic layers
2. Mock all the repositories using a mocking framework.

**Stage: Micro-service implementation**

1. Create a API project which references the business logic layer created earlier
2. Implement service documentation using swagger
3. All exceptions in the micro-service must be handled and logged using a logging library
4. Create the following end-points and test them using postman and export the requests into a json file.

Table 2 : Account Service - Endpoint - 1

|  |  |
| --- | --- |
| **URL** | /api/grades |
| **Request Type** | GET |
| **User Role** | HR |
| **Trigger** | Front end |
| **Description** | HRs of the system will use this endpoint to see the list of available grades which can be used for employees |
| **Inputs** |  |
| **Outputs** | GradeDTO |

Table 3 : Account Service - Endpoint - 2

|  |  |
| --- | --- |
| **URL** | /api/employees |
| **Request Type** | GET |
| **User Role** | HR |
| **Trigger** | Front end |
| **Description** | Using this endpoint an HR will be able to fetch details of all employees present in the system |
| **Inputs** |  |
| **Outputs** | EmployeeDTOs |

Table 4 : Account Service - Endpoint - 3

|  |  |
| --- | --- |
| **URL** | /api/employees/<id> |
| **Request Type** | GET |
| **User Role** | HR |
| **Trigger** | Front end |
| **Description** | The HRs will use this endpoint to fetch details of a single employee |
| **Inputs** | EmployeeId |
| **Outputs** | EmployeeDTO |

Table 5 : Account Service - Endpoint - 4

|  |  |
| --- | --- |
| **URL** | /api/employees |
| **Request Type** | POST |
| **User Role** | HR |
| **Trigger** | Front end |
| **Description** | With the help of this endpoint the HRs will be able to add new employees into the system |
| **Inputs** | EmployeeDTO |
| **Outputs** | Status code |

Table 6 : Account Service - Endpoint - 5

|  |  |
| --- | --- |
| **URL** | /api/employees/<id> |
| **Request Type** | PUT |
| **User Role** | HR |
| **Trigger** | Front end |
| **Description** | This endpoint will allow the HRs to update the grade of an employee |
| **Inputs** | EmployeeId, EmployeeDTO |
| **Outputs** | Status code |

Table 7 : Account Service - Endpoint - 6

|  |  |
| --- | --- |
| **URL** | /api/employees/<id> |
| **Request Type** | DELETE |
| **User Role** | HR |
| **Trigger** | Front end |
| **Description** | Using this endpoint the HRs can delete an employee from the system |
| **Inputs** | EmployeeId |
| **Outputs** | Status code |

**Stage: Font end design**

Create the following components as per the specification provided below.

1. AddEmployeeComponent
2. Create an Add employee component which will be used by HR to add new employees with the system
3. Define a navigation to the component in the application navbar
4. Once all details are validated the form should be allowed to be submitted and an acknowledgement must be displayed.
5. Role of user should be displayed in the form of radio buttons
6. The grade of the employee should be selected from a dropdown list
7. EmployeeListComponent
8. Develop a component for HRs to be accessed from the navigation bar of the application
9. The component should list all the employees details in a tabular format.
10. Each record in table should have a delete button
11. Upon clicking the delete button, take a confirmation from the user.
12. Once user confirm, the employee record must be delete from the system and an acknowledgement should be displayed.
13. ChangeGradeComponent
14. Create a new component to update the grade of an employee.
15. The component should be accessible to HR from the application menu
16. Provide a textbox to the user to type the employee id and search the employee
17. Only the grade of the employee should be editable.
18. Once the grade is updated in the system, display an acknowledgement.

**Stage: Integration of Front end and back end**

1. Create a data service in the font-end application which will communicate with the micro services.
2. Use the data service in the components to make them interact with the API
3. Valid error messages should be shown based on various response status codes received form the API

### **Module – Travel Planner**

This module will provide the following features to the employees and the HR

1. An employee can raise a new travel request and obtain an auto-generated travel request id.
2. A HR can approve/reject a travel request. If a travel request is approved then the HR should also provided the assigned budget.
3. An employee can track their travel request status

**Stage: Database Implementation**

1. Design a data base as per the following ER diagram provided.

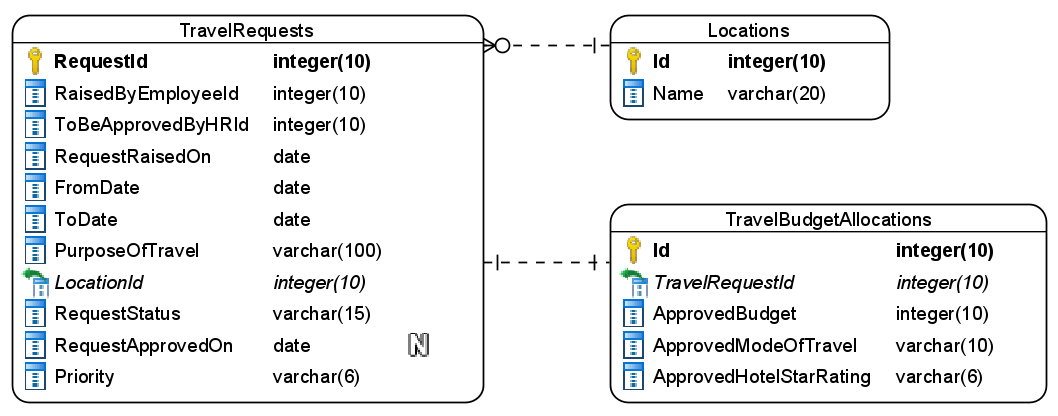


Figure 4 : ER Diagram – Travel Planner

1. Apply the following constraints apart from primary keys and foreign keys on the database
   1. The ToDate should not be lesser than from date.
   2. RequestRaisedOn should be taken by default as current date
   3. Request status should accept on – New, Approved, rejected
   4. Allowed values for ApprovedModeOfTravel must be – Air/Train/Bus
   5. Values allowed for ApprovedHotelStarRating is 3-Star/5-Star/7-Star

**Note**: The data in location table shouldbe auto-populated on the application startup.

**Stage: Data Access Layer Design**

1. Create a library project and add ORM support into it.
2. Use the ORM to map the entities to database as per the ER diagram provided.
3. Use repository per entity pattern and generate the repositories to perform the following operations
   1. Return all locations
   2. Insert a new travel request
   3. Return all pending requests for HR id
   4. Return a request by request id
   5. Reject request
   6. Approve request

**Stage: Business Logic Layer Development**

1. Develop a library which reference the Data Access Library project created earlier
2. This class library will contain various service classes which will encapsulate the business logic for the application.
3. Use dependency injection to in service classes to inject the required repositories.
4. Create the service classes using the single responsibility principle which perform the given operations as follows
   1. Get all locations
   2. Add new travel request an obtain a travel request id
   3. Track a travel request
   4. Get new travel request for a HR
   5. Approve/reject travel request
   6. Calculate allowed budget
5. Following business rules must be implemented as part of the business service class
   1. Total number of days an employee is allowed to be on-site is as follows
      1. Priority 1 – maximum 30 days
      2. Priority 2 – maximum 20 days
      3. Priority 3 – maximum 10 days
   2. Use the following calculations for allowed budget limit
      1. Grade 1 employees – maximum per day 10000 INR
      2. Grade 2 employees – maxium per day 12500 INR
      3. Grade 3 employees – maximum per day 15000 INR
   3. A HR is allowed a 5-star or 7-star hotel where as other employees are allowed to have a 3-star or 5-star hotel for their stay

**Stage: Unit Testing**

1. Create a new Unit test project to test the service classes created in business logic layers
2. Mock all the repositories using a mocking framework.

**Stage: Micro-service implementation**

1. Create a API project which references the business logic layer created earlier
2. Implement service documentation using swagger
3. All exceptions in the micro-service must be handled and logged using a logging library
4. Create the following end-points and test them using postman and export the requests into a json file.

Table 9 : Travel Planner - Endpoint - 1

|  |  |
| --- | --- |
| **URL** | /api/travelrequests/locations |
| **Request Type** | GET |
| **User Role** | Employees, HRs, TravelDeskExec |
| **Trigger** | Front end |
| **Description** | This endpoint will provide employees with a list of locations which they can select while creating a new travel request |
| **Inputs** |  |
| **Outputs** | LocationDTOs |

Table 10 : Travel Planner - Endpoint - 2

|  |  |
| --- | --- |
| **URL** | /api/travelrequests/new |
| **Request Type** | POST |
| **User Role** | Employees |
| **Trigger** | Front end |
| **Description** | Using this endpoint the employees can create a new travel request |
| **Inputs** | TravelRequestDTO |
| **Outputs** | Status code and new travel request id |

Table 11 : Travel Planner - Endpoint - 3

|  |  |
| --- | --- |
| **URL** | /api/travelrequests/<HRid>/pending |
| **Request Type** | GET |
| **User Role** | HRs |
| **Trigger** | Front end |
| **Description** | A HR will be able to fetch a list of newly created travel request which needs to be approved by him/her |
| **Inputs** | HRId |
| **Outputs** | TravelRequestDTOs |

Table 12 : Travel Planner - Endpoint - 4

|  |  |
| --- | --- |
| **URL** | /api/travelrequests/<trid> |
| **Request Type** | GET |
| **User Role** | Employee, HR, TravelDeskExec |
| **Trigger** | Front end, other micro-services |
| **Description** | The endpoint will be responsible for fetching and displaying the details of a travel request raise earlier. If request is approved it should also include the approval details |
| **Inputs** | TravelRequestId |
| **Outputs** | TravelRequestDetailsDTO or status code |

Table 13 : Travel Planner - Endpoint - 5

|  |  |
| --- | --- |
| **URL** | /api/travelrequests/<trid>/update |
| **Request Type** | PUT |
| **User Role** | HR |
| **Trigger** | Front end |
| **Description** | A HR will use this endpoint to approve or reject the travel request raised by his employees |
| **Inputs** | TravelRequestId and UpdateTravelRequestDTO |
| **Outputs** | Status code |

Table 14 : Travel Planner - Endpoint - 6

|  |  |
| --- | --- |
| **URL** | /api/travelrequests/calculatebudget |
| **Request Type** | POST |
| **User Role** | HR |
| **Trigger** | Front end |
| **Description** | This endpoint will be responsible for calculating and returning the applicable budget for the travel |
| **Inputs** | TravelRequestDTO |
| **Outputs** | Applicable budget |

**Stage: Font end design**

Create the following components as per the specification provided below.

1. NewTravelRequestComponent
2. Create a component which contains a form to allow employees to create a new travel request.
3. The form should accept the details like FromDate, ToDate, Location, Purpose and priority
4. Use a dropdown list for the location
5. Use a set of radio buttons for the priority
6. The HR id must be auto-populated and displayed to user in a readonly textbox
7. Once all details are validated then form must be allowed to be submitted and acknowledgment must be displayed along with the generated TravelRequestId
8. Define a navigation link to the component available only for all the employees, traveldeskexec and HRs
9. NewRequestListComponent
   1. Create a component which will be used by the HR to view all the new request which are tagged to him
   2. Display the request in a tabular format.
   3. Each row must have 2 buttons to approve or reject a travel request
   4. If HR selects approve travel request then redirect him to travelrequest details component by passing the travel request id as route parameter.
   5. Create a navigate to the component such that it is accessible to the HRs.
10. TravelRequestDetailsComponent
    1. Develop a component which can be used by HRs and employees
    2. If an employee access the component then he/she must enter travel request id and click the search button to view the details of travel request
    3. If request is approved then the budget and other details should also be displayed.
    4. When a HR navigates to the component from NewRequestListComponent by clicking the approve button then travel request details must be automatically fetched and displayed to him.
    5. A HR should also get a dropdown to choose the mode of travel and hotel star rating. The travel budget must be calculated and displayed automatically.
    6. Once all the details are validated the HR should be able to submit the approval and get an acknowledgement.
    7. Provide a navigation to the component in the application menu for the employees.
    8. Provide a switch on the component to change the view from HR to employees and vice-versa

**Stage: Integration of Front end and back end**

1. Create a data service in the font-end application which will communicate with the micro services.
2. Use the data service in the components to make them interact with the API
3. Valid error messages should be shown based on various response status codes received form the API

### **Module – Reservations management**

This module will provide the following services

1. TravelDeskExec can upload the reservations of hotel, car and flight/train/bus for the purpose of employee travel
2. An employee will be able to track the reservations done for him/her
3. Employees can also download the uploaded reservation documents

**Stage: Database Implementation**

1. Design a data base as per the following ER diagram provided.

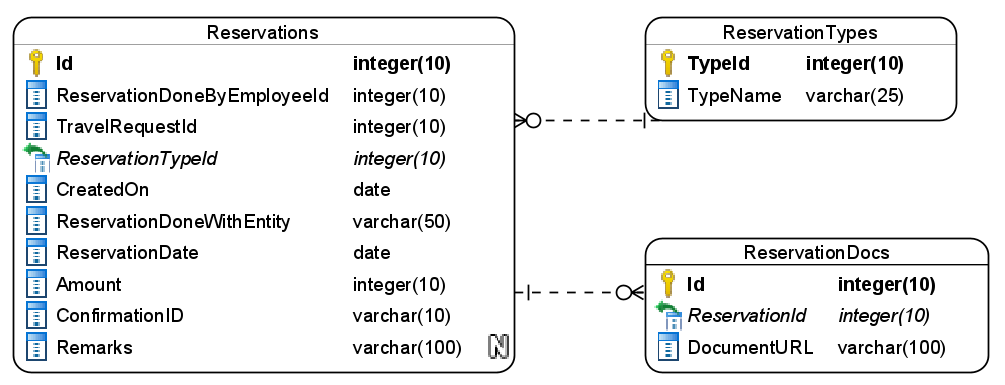


Figure 5 : ER Diagram - Reservation Management

1. Apart from primary and foreing keys implement the following additional constraints
   1. CreatedOn must be automatically taken as current date.
   2. The amount must be a positive number only

Note: Pre-populate the reservation types as flight/train/bus/cab/hotel during the application startup

**Stage: Data Access Layer Design**

1. Create a library project and add ORM support into it.
2. Use the ORM to map the entities to database as per the ER diagram provided.
3. Use repository per entity pattern and generate the repositories to perform the following operations
   1. Return reservation types
   2. Insert new reservation
   3. Return reservations by travel request id

**Stage: Business Logic Layer Development**

1. Develop a library which reference the Data Access Library project created earlier
2. This class library will contain various service classes which will encapsulate the business logic for the application.
3. Use dependency injection to in service classes to inject the required repositories.
4. Create the service classes following the single responsibility principle which perform the given operations as follows
   1. Add a new reservation
   2. Download a reservation document
   3. Get reservations by travel request id
   4. Get reservation types
5. Following business rules must be implemented as part of the business service class
   1. ReservationDate for a train/bus reservation must be 1 day before the fromdate mentioned in Travel Planner. Communicate with travel planner service to get the details
   2. ReservationDate for hotel must be same as the from date in travel plan.
   3. There must be exactly 3 reservations per travel, one for flight/bus/train, one for hotel and one for cab travel to hotel from flight/bus/train
   4. The amount of all 3 reservations must not exceed the 70% of the allocated budget
   5. Each reservation document must be in pdf format with a maximum size of 1MB. If the size limit exceeds then a user defined exception “DocumentSizeLimitExceededException” must be raised.

**Stage: Unit Testing**

1. Create a new Unit test project to test the service classes created in business logic layers
2. Mock all the repositories using a mocking framework.

**Stage: Micro-service implementation**

1. Create a API project which references the business logic layer created earlier
2. Implement service documentation using swagger
3. All the exceptions must be handled and logged using a logging library.
4. Create the following end-points and test them using postman and export the requests into a json file.
5. This micro-service will also communicate with Travel Planner Service to get details of the vehicle for calculation of fares and max passenger limit

Table 16 : Reservation Management - End point - 1

|  |  |
| --- | --- |
| **URL** | /api/reservations/types |
| **Request Type** | GET |
| **User Role** | TravelDeskExec |
| **Trigger** | Front end |
| **Description** | An travel desk executive should be able to choose the type of reservation documents he/she is uploading |
| **Inputs** |  |
| **Outputs** | ReservationTypeDTOs |

Table 17 : Reservation Management - End point - 2

|  |  |
| --- | --- |
| **URL** | /api/reservations/add |
| **Request Type** | POST |
| **User Role** | TravelDeskExec |
| **Trigger** | Front end |
| **Description** | Using the endpoint a user will be add the new reservation details along with the PDF copy of the reservation document obtain from the entity like airline |
| **Inputs** | AddReservationDTO |
| **Outputs** | Status code |

Table 18 : Reservation Management - End point - 3

|  |  |
| --- | --- |
| **URL** | /api/reservations/track/<travelrequestid> |
| **Request Type** | GET |
| **User Role** | Employees |
| **Trigger** | Front end and other micro services |
| **Description** | Employees will be able to use this endpoint to see the details of the reservations made for them |
| **Inputs** | TravelRequestId |
| **Outputs** | ReservationDTOs |

Table 19 : Reservation Management - End point - 4

|  |  |
| --- | --- |
| **URL** | /api/reservations/<reservationid> |
| **Request Type** | GET |
| **User Role** | Employees |
| **Trigger** | Front end |
| **Description** | This endpoint will allow the employees to view the reservation details |
| **Inputs** | ReservationID |
| **Outputs** | ReservationDTO |

Table 20 : Reservation Management - End point - 5

|  |  |
| --- | --- |
| **URL** | /api/reservations/<reservationid>/download |
| **Request Type** | GET |
| **User Role** | Employees |
| **Trigger** | Front end |
| **Description** | This endpoint will allow the employees to download the attached document uploaded along with reservation |
| **Inputs** | ReservationID |
| **Outputs** | File |

**Stage: Font end design**

Create the following components as per the specification provided below.

1. UploadReservationComponent
2. Create a component which will allow the travel desk executives to create a upload the details of a reservation made for an employee
3. Provide a dropdown list for the reservation type
4. Once all details are validated then the form must be submitted and an acknowledgement should be displayed
5. Provide a navigation to component for travel desk executives via navbar
6. TrackReservationComponent
7. Design a search track reservation component which should be accessed by employees using the navigation bar
8. Component should provide the search facility to employee to type their travel request id
9. All reservations for the given travel request ID must be displayed in bootstrap cards format.
10. Each record should also have a view details button to view details which will redirect to ReservationDetailsComponent by passing in the reservation id
11. ReservationDetailsComponent
    1. Develop a Reservation details component which can be navigated to from track Reservation component
    2. It should display a single reservation details along with a download button
    3. Once user clicks on the download button the associated document uploaded by the travel desk executive must be download on to the users system.

**Stage: Integration of Front end and back end**

1. Create a data service in the font-end application which will communicate with the micro services.
2. Use the data service in the components to make them interact with the API
3. Valid error messages should be shown based on various response status codes received form the API

### **Module – Reimbursement management**

Once an employee is returned from their travel they will use this module to claim reimbursements for various expenses like food, daily transportation and laundry service which they utilized during their course of stay on the travel location. This module will provide the following facilities

1. It will allow the employees to create a new Reimbursement request
2. A travel desk executive can process the request by either accepting it or rejecting it
3. The HRs should be able to see the expenses incurred by the employee under various expense types in the form of a report

**Stage: Database Implementation**

1. Design a data base as per the following ER diagram provided.

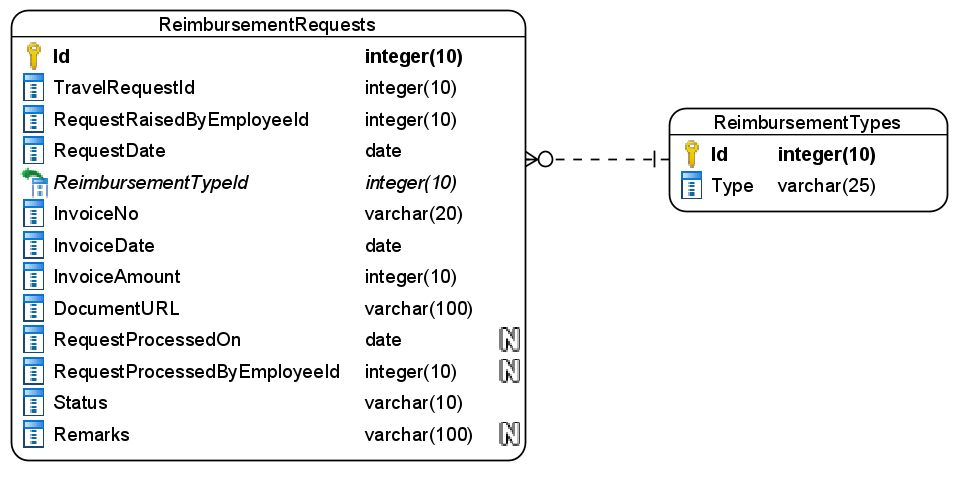


Figure 6 : ER Diagram - Reimbursement Service

1. Enfore the following constraints along with primary and foreign keys
   1. Request date must be automatically taken as current date
   2. Allowed values for the status is – new/approved/rejected
   3. ProcessedOn must not be a past date
   4. Invoice date must be a passed date

**Note**: Feed the data into the ReimbursementTypes table when the application starts. Values can be Food/Water/Laundry/LocalTravel

**Stage: Data Access Layer Design**

1. Create a library project and add ORM support into it.
2. Use the ORM to map the entities to database as per the ER diagram provided.
3. Use repository per entity pattern and generate the repositories to perform the following operations
   1. Return the available Reimbursement types
   2. Insert a new Reimbursement
   3. Return all Reimbursements by travel request id
   4. Process the reimbursement
   5. Return a reimbursement by id

**Stage: Business Logic Layer Development**

1. Develop a library which reference the Data Access Library project created earlier
2. This class library will contain various service classes which will encapsulate the business logic for the application.
3. Use dependency injection to in service classes to inject the required repositories.
4. Create the service classes following the single responsibility principle which perform the given operations as follows
   1. Get the Reimbursement types
   2. Get all reimbursements for a travel request id
   3. Process a reimbursement
   4. Get a reimbursement by id
   5. Add new reimbursement
5. Following business rules must be implemented as part of the business service class
   1. Allowed documents for uploading is pdf with the size of 256kb
   2. Invoice date must be with in the from and to date of the travel which can be obtained from travel planner
   3. If a reimbursement is rejected then a remarks must be provided.
   4. The total allowed expense per day for food, laundry and local travel is as follows
      1. Food and water – 1000-1500 per day
      2. Laundry – 250-500 per day
      3. Local travel – upto 1000 per day

**Stage: Unit Testing**

1. Create a new Unit test project to test the service classes created in business logic layers
2. Mock all the repositories using a mocking framework.

**Stage: Micro-service implementation**

1. Create a API project which references the business logic layer created earlier
2. Implement service documentation using swagger
3. Create the following end-points and test them using postman and export the requests into a json file.

Table 22 : Reimbursement Management - End point - 1

|  |  |
| --- | --- |
| **URL** | /api/reimbursements/types |
| **Request Type** | GET |
| **User Role** | Employees |
| **Trigger** | Front end |
| **Description** | This end points will provide the available reimbursement types in the system |
| **Inputs** |  |
| **Outputs** | ReimbursementTypeDTOs |

Table 23 : Reimbursement Management - End point - 2

|  |  |
| --- | --- |
| **URL** | /api/reimbursements/add |
| **Request Type** | POST |
| **User Role** | Employees |
| **Trigger** | Front end |
| **Description** | Using this end-point employees will be able to add request for a reimbursement |
| **Inputs** | NewReimbursementRequestDTO |
| **Outputs** | Status code |

Table 24 : Reimbursement Management - End point - 3

|  |  |
| --- | --- |
| **URL** | /api/reimbursements/<travelrequestid>/requests |
| **Request Type** | GET |
| **User Role** | Employees, TravelDeskExec |
| **Trigger** | Front end |
| **Description** | This endpoint will return details of all the reimbursement requests raised for a given travel request id |
| **Inputs** | TravelRequestId |
| **Outputs** | ReimbursementDTOs |

Table 25 : Reimbursement Management - End point - 4

|  |  |
| --- | --- |
| **URL** | /api/reimbursements/<reimbursementid> |
| **Request Type** | GET |
| **User Role** | TravelDeskExec |
| **Trigger** | Front end |
| **Description** | Travel desk executives can use this endpoint to view detail of a single reimbursement request |
| **Inputs** | ReimbursementId |
| **Outputs** | ReimbursementDTO |

Table 26 : Reimbursement Management - End point - 5

|  |  |
| --- | --- |
| **URL** | /api/reimbursements/<reimbursementid>/process |
| **Request Type** | PUT |
| **User Role** | TravelDeskExec |
| **Trigger** | Front end |
| **Description** | Using this endpoint the travel desk executive will be able to accept or reject a given reimbursement |
| **Inputs** | ProcessReimbursementDTO |
| **Outputs** | Status code |

**Stage: Font end design**

Create the following components as per the specification provided below.

1. NewReimbursementComponent
2. Create a component and allow the navigation to it for employees which contains a form
3. The form should have a dropdown list to choose the type of reimbursement the user is submitting
4. Validate all the data before it’s submitted
5. On successful submission of Reimbursement display an acknowledgement.
6. ReimbursementsListComponent
7. Create a component which is accessible to travel desk executive, employees and HRs
8. It must contain a textbox to accept a travel request id on submission of which it should fetch the reimbursements raised
9. For employees simply display the reimbursements in a tabular format
10. For HR display the details a summary of various reimbursement categories and their amount along with total
11. For Travel desk executive display the reimbursements in a table along with a process button. Once the button is clicked it should redirect to process reimbursement component by passing the reimbursement id as parameter
12. ProcessReimbursementComponent
13. Design a new component for processing Reimbursement filed by employees
14. Component must be accessible to travel desk executives
15. It should display the detail of a single reimbursement request along with a remark textbox and accept and reject buttons.
16. If a reimbursement request is rejected then the remarks is mandatory.
17. Once the form is successfully submitted display an acknowledgement.

**Stage: Integration of Front end and back end**

1. Create a data service in the font-end application which will communicate with the micro services.
2. Use the data service in the components to make them interact with the API
3. Valid error messages should be shown based on various response status codes received form the API

# Deployment requirements

1. All the Microservices must be deployed on a local web server like IIS or Apache Tomcat
2. All the Microservices must be independently deployable.
3. These services must be consumed from an Front end app running in a local environment.

# Design Considerations

Java and Dotnet specific design considerations are attached here. These design specifications, technology features have to be strictly adhered to.



Refer this link for the coding standards. <https://cognizantonline.sharepoint.com/:w:/r/sites/GTP-Solutions/Gencsharepath/Shared%20Documents/Internship2020/FSE/Coding%20standards/Effective%20coding%20standards.docx?d=w6430574d9db5478bbbe37c25b16e68e2&csf=1&web=1&e=84lTVf>

### Most Important and Common rules

|  |  |
| --- | --- |
| **Category** | **Rule** |
| Database | Table names in database must be pascal cased and plural. All primary keys must be named as Pk\_<table>. All foreign keys must be named as FK\_<PrimaryKeyTable>\_<ForeignKeyTable> |
| Database | Column names must be pascal cased. Multi-word column must be split using \_ (underscore) |
| Coding | Follow pascal casing for naming classes, interfaces, methods, properties and other public members |
| Coding | Use camel casing for method parameter name, backing fields for properties and private variables. Consts must be capitalized |
| Coding | All exceptions must be handled and logged using a logging library |
| Coding | For communication between micro-services use the HttpClient class available in .Net and Java |
| Unit testing | Each method in services classes in business logic must be unit tested using nUnit/jUnit |
| Unit testing | Use a mocking library to mock the repositories while performing tests for business logic layer |
| Code Coverage | Should be minimum 90% |
| Front end(Angular/React ONLY) | Use pascal casing for the component names |
| Front end(Angular/React ONLY) | Create all components and data services in Angular/React project in dedicated folders |
| GitHub | Create ONLY Private Repositories.  No password should be stored.  DO NOT Mention in the Profile that You work for Cognizant |

Any deviation from the high level design must be approved by trainer, mentor and solutions team

# Reference learning

Please go through all of these k-point videos for

Microservices deployment into Azure Kubernetes Service.

|  |
| --- |
| [AzureWithCICD-1](https://cognizant.kpoint.com/app/video/gcc-19532393-d4e0-4fd9-8a0c-80ecbdb349d3) |
| [AzureWithCICD-2](https://cognizant.kpoint.com/app/video/gcc-6633a958-ab72-4c69-b926-fe832e4b56a1) |
| [AzureWithCICD-3](https://cognizant.kpoint.com/app/video/gcc-553eb186-c1cf-448e-96fc-a96fe37b2e6a) |
| [AzureWithCICD-4](https://cognizant.kpoint.com/app/video/gcc-fad7d4af-d651-4501-99c6-2785190670c2) |

**Other References:**

|  |  |
| --- | --- |
| Java 8 Parallel Programming | <https://dzone.com/articles/parallel-and-asynchronous-programming-in-java-8> |
| Feign client | [https://dzone.com/articles/Microservices-communication-feign-as-rest-client](https://dzone.com/articles/microservices-communication-feign-as-rest-client) |
| Swagger (Optional) | [https://dzone.com/articles/centralized-documentation-in-Microservice-spring-b](https://dzone.com/articles/centralized-documentation-in-microservice-spring-b) |
| ECL Emma Code Coverage | <https://www.eclipse.org/community/eclipse_newsletter/2015/august/article1.php> |
| Lombok Logging | <https://javabydeveloper.com/lombok-slf4j-examples/> |
| Spring Security | <https://dzone.com/articles/spring-boot-security-json-web-tokenjwt-hello-world> |
| H2 In-memory Database | <https://dzone.com/articles/spring-data-jpa-with-an-embedded-database-and-spring-boot>  <https://www.baeldung.com/spring-boot-h2-database> |
| AppInsights logging | <https://www.codeproject.com/Tips/1044948/Logging-with-ApplicationInsights> |
| Error response in WebApi | <https://stackoverflow.com/questions/10732644/best-practice-to-return-errors-in-asp-net-web-api> |
| Read content from CSV | <https://stackoverflow.com/questions/26790477/read-csv-to-list-of-objects> |
| Access app settings key from appSettings.json in .Net core application | <https://www.c-sharpcorner.com/article/reading-values-from-appsettings-json-in-asp-net-core/>  <https://docs.microsoft.com/en-us/aspnet/core/fundamentals/configuration/?view=aspnetcore-3.1> |

# Project Templates









# Change Log

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Changes Made | | | |
| V1.0.0 | Initial baseline created on 07-September-2022 by Khaleelullah Hussaini Syed | | | |
| V1.0.1 |  | | | |
| **Section No.** | **Changed By** | **Effective Date** | **Changes Effected** |
| 2.3 | Khaleelullah Hussaini Syed | 18 October 2022 | Updated project scope with new modules |
| 3.0 | Updated use case diagram |
| 4.0 | New architecture diagram |
| 5.0 | Replaced API gateway phase with deployment |
| 6.1.1 | Replaced the authentication module with Human resources module |
| 6.1.2 | Updated database diagram and user roles, removed API gateway |
| 6.1.3 | Removed API gateway |
| 6.1.4 | Removed API gateway |
| 8 | Updated design consideration and ppt |
| 10 | Added new section on project templates for .net back end, react front end and angular front end |