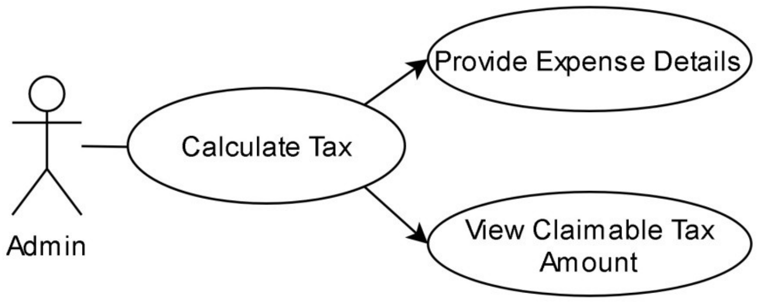
Tax Management System

1.0 Introduction

1.1 Functional Requirements

The Tax Management System is an application used by the client who is having a group of companies which provide various services like transport, food, accommodation, hospitals, institutes, and so on. The client had offered a benefit to their employees as, if an employee uses the services provided by their own group of companies, a claim of the tax amount will be provided as reimbursement to the employee. The accounts administrator needs to manually calculate the tax claim amount from each employee based on the expense type and expense amount.
To automate this calculation process, develop a Spring boot, Spring MVC application. This application needs to get employee id, expense amount and expense type as input. Based on the business rules given below, the claim amount should be calculated and displayed. The application will be used by the account's admin of the company.

1.2. Use Case Diagram



2.0 Technical Specifications

2.1 Process Flow

- The first page of the application is taxclaim.jsp page, this page should have the below form components.

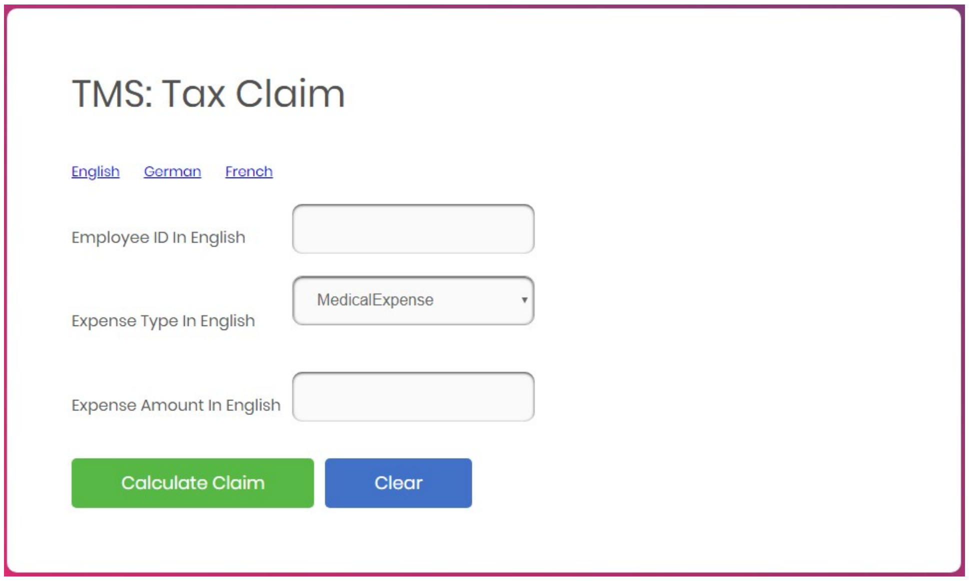
Component	Component Type	Component ID
Employee ID	Textbox	employeeId
Expense Type	Drop down	expenseType
Expense Amount	Textbox	expenseAmount
Calculate Claim	Submit	calculateClaim (should be name and not id)

- In the same page we have the provision to choose language English, German and French. By default, the page content will be displayed in English. If we choose the language as French the label content and the error message in the page should displayed in French and if we choose the language as German the label content and the error message in the page should displayed in German. Ensure you implement the concept of Internationalization.

- In the taxclaim.jsp page, ensure you use the component type and id correctly as specified in the above table.
- Assumption: The expense amount entered will be in rupees always.
- The values in the Expense Type drop-down must be auto populated from the controller with the values as given in the below table. They should not be populated/hardcoded inside the JSP.

Expense Type
MedicalExpense
TravelExpense
FoodExpense

- Sample taxclaim.jsp for displaying the page content in English



TMS: Tax Claim

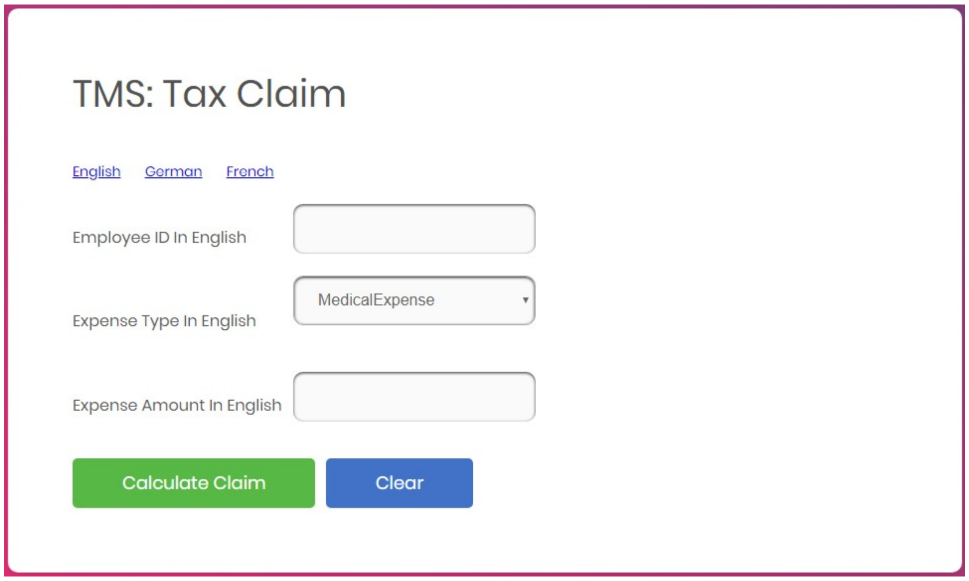
English German French

Employee ID In English

Expense Type In English MedicalExpense

Expense Amount In English

- Sample taxclaim.jsp for displaying the page content in German



TMS: Tax Claim

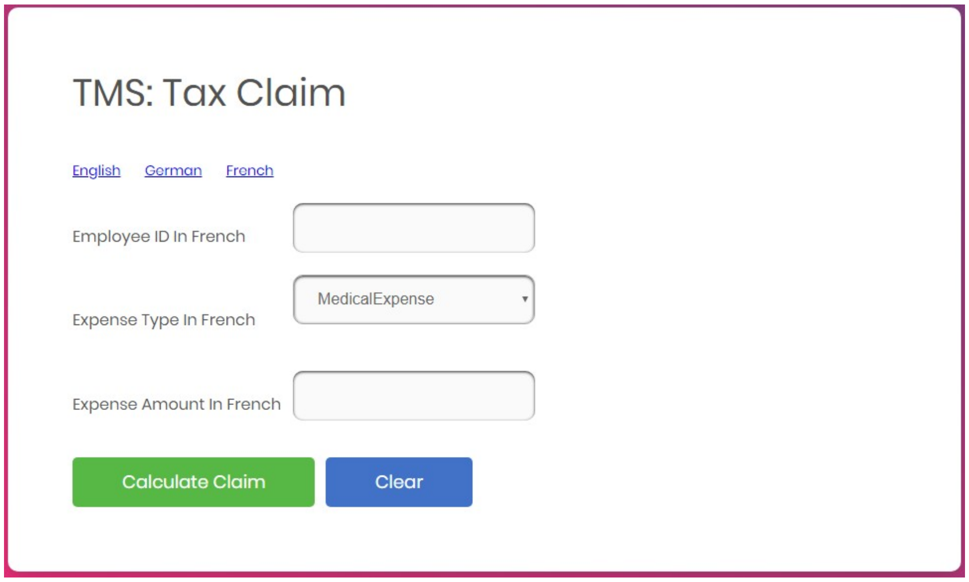
English German French

Employee ID In English

Expense Type In English MedicalExpense

Expense Amount In English

- Sample taxclaim.jsp for displaying the page content in French

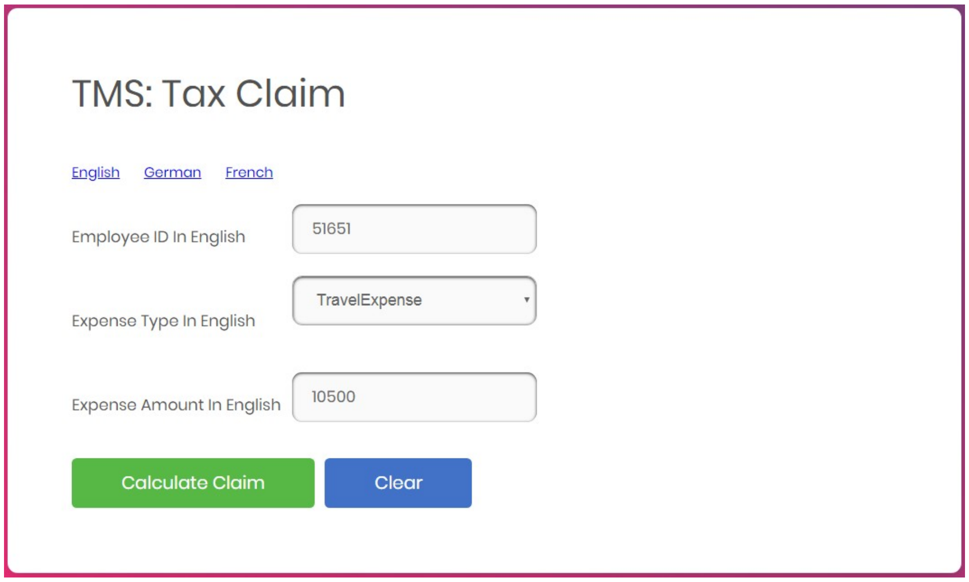


TMS: Tax Claim

English German French

Employee ID In French	
Expense Type In French	MedicalExpense
Expense Amount In French	

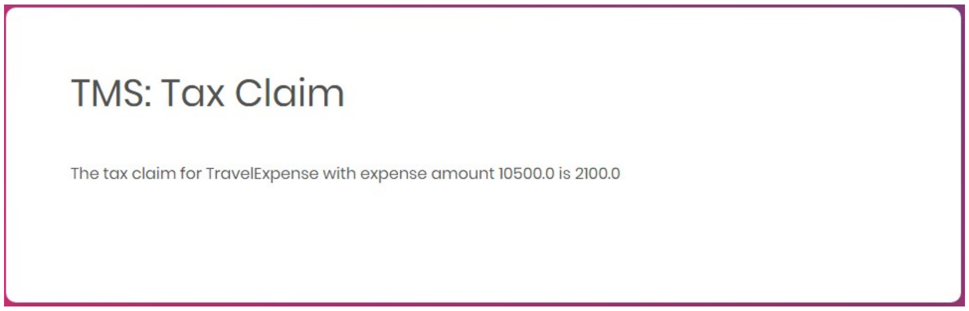
- The Calculate Claim button, on-click after all successful validations (Refer 2.2 for validation), must generate the tax claim amount based on the expense type and expense amount in the result.jsp file. In this page the message should be displayed as "The tax claim for <<expenseType>> with expense amount <<expenseAmount>> is <<taxClaimAmount>>". The result should be rendered inside the <h2> tag. Refer 2.3 for the logic to be followed for doing calculation.



TMS: Tax Claim

English German French

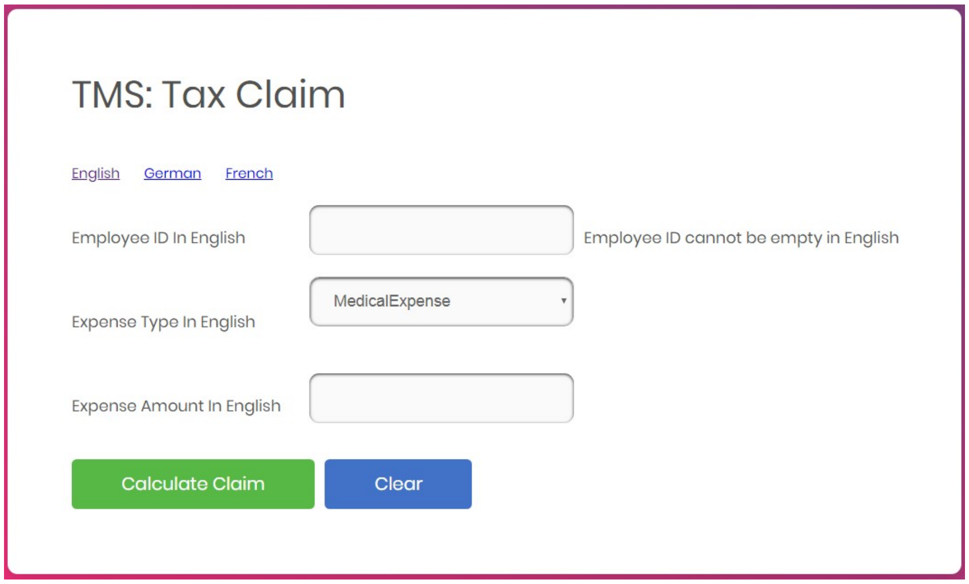
Employee ID In English	51651
Expense Type In English	TravelExpense
Expense Amount In English	10500


TMS: Tax Claim

The tax claim for TravelExpense with expense amount 10500.0 is 2100.0

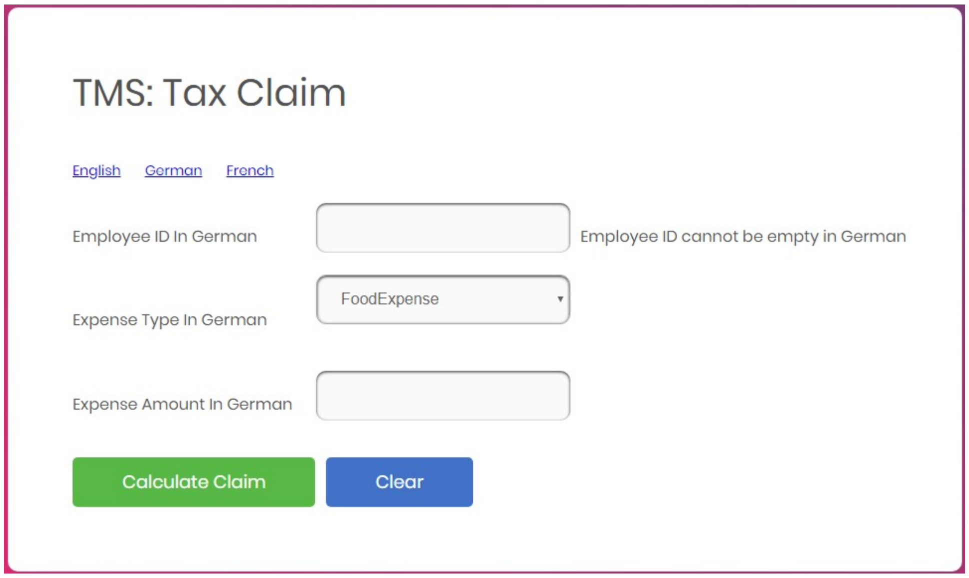
2.2 Business Validation

- Rule: Employee ID field is mandatory.
- Message: Employee ID cannot be empty in <<locale>>
    - If validation fails, UI should display the error message - "Employee ID cannot be empty in <<locale>>".


TMS: Tax Claim

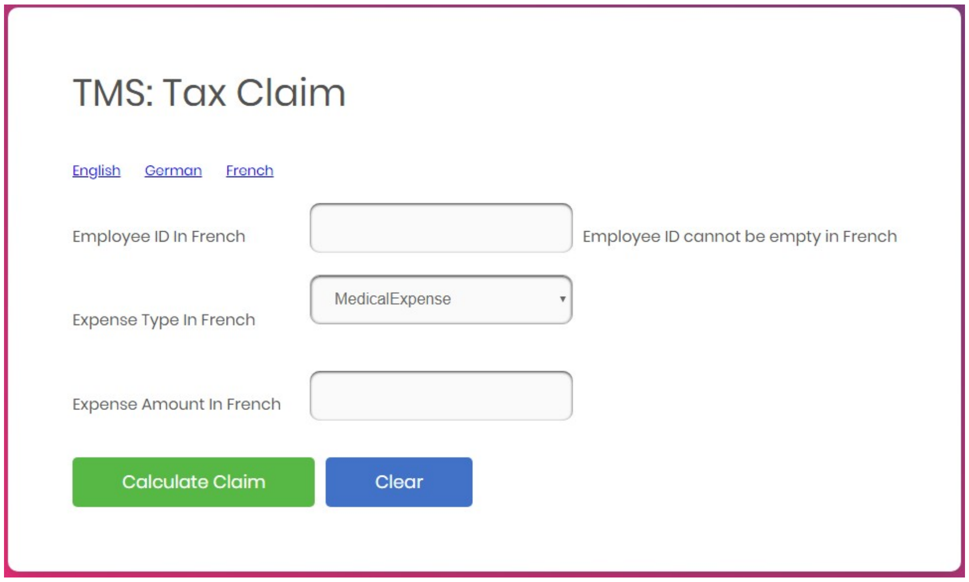
English German French

Employee ID In English		Employee ID cannot be empty in English
Expense Type In English	MedicalExpense	
Expense Amount In English		


TMS: Tax Claim

English German French

Employee ID In German		Employee ID cannot be empty in German
Expense Type In German	FoodExpense	
Expense Amount In German		


TMS: Tax Claim

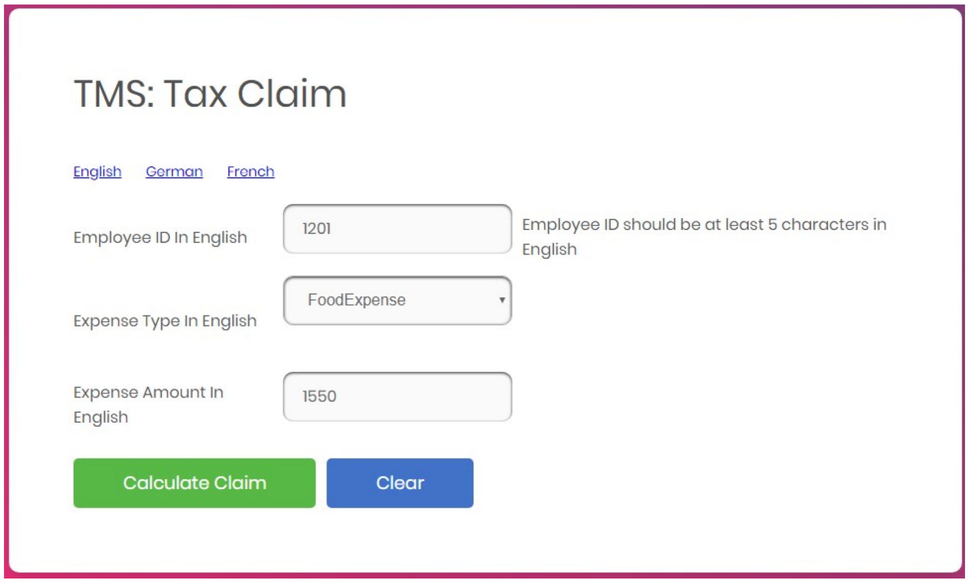
English German French

Employee ID In French — Employee ID cannot be empty in French

Expense Type In French: MedicalExpense

Expense Amount In French

- <u>Rule:</u> Employee ID should be at least 5 characters
- <u>Message:</u> Employee ID should be at least 5 characters in <<locale>>
  - If validation fails, UI should display the error message - "Employee ID should be at least 5 characters".


TMS: Tax Claim

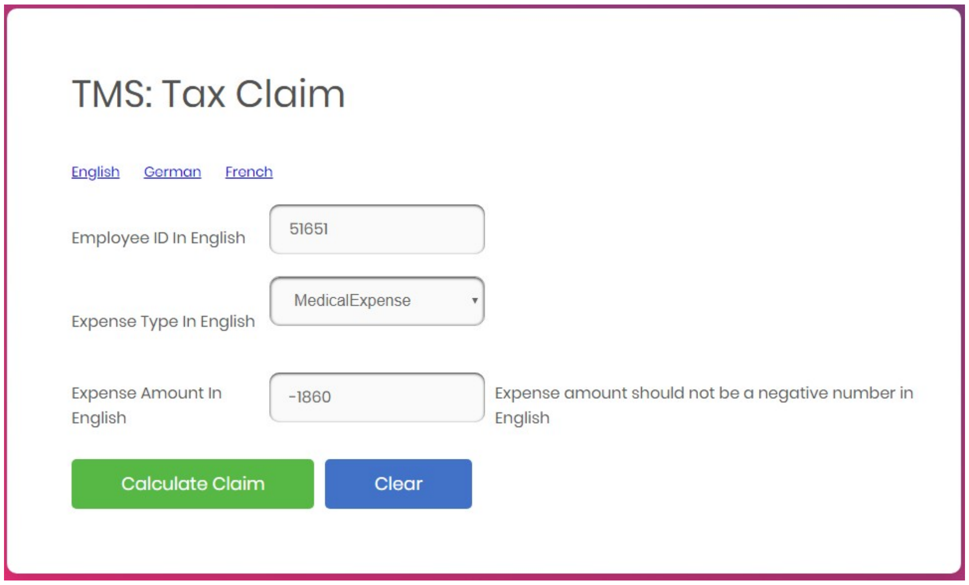
English German French

Employee ID In English: 1201 — Employee ID should be at least 5 characters in English

Expense Type In English: FoodExpense

Expense Amount In English: 1550

- <u>**Rule:**</u> Expense Amount should not be a negative number.
- <u>**Message:**</u> Expense amount should not be a negative number in <<locale>>
  - If the validation fails, UI should display the error message - "Expense amount should not be a negative number in <<locale>>"



TMS: Tax Claim

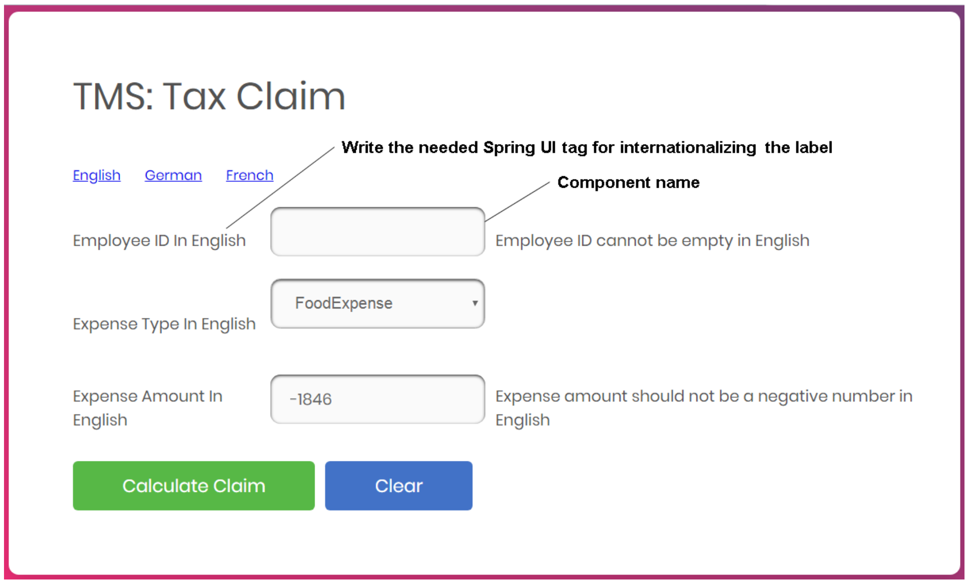
English German French

Employee ID In English: 51651

Expense Type In English: MedicalExpense

Expense Amount In English: -1860 — Expense amount should not be a negative number in English

- **Note:** The messages have to be retrieved from the appropriate properties file. The Property file has been given as part of code skeleton.
- The table along with the tr and td tags are provided as part of the code skeleton. Each td is provided with the id attribute. Do not change the id value which is provided as part of the code skeleton. Include the required Spring UI tag for Internationalization in the first column, the corresponding component in the second column and the Spring tag for Internationalization to display the error message in the third column for all the rows as shown below.



TMS: Tax Claim

English German French

Employee ID In English: — Employee ID cannot be empty in English

Expense Type In English: FoodExpense

Expense Amount In English: -1846 — Expense amount should not be a negative number in English

Write the needed Spring UI tag for internationalizing the label
Component name
Calculate Claim
Clear

2.2 Control Flow

- By giving the request /getTaxClaimFormPage from the browser (Eg: http://localhost/8080/getTaxClaimFormPage), the admin should be redirected to the Tax Claim page which is taxclaim.jsp
- Admin will enter the employee id, choose the expense type and enter expense amount.

- On clicking calculate claim button, the expense amount should be validated as per the business rules & validations.
- The tax claim amount should be calculated and displayed in the page is result.jsp

2.3 Business Rules

- Tax paid for each expense type is given in the table below.

Expense Type	**Tax percentage based on Expense Amount Range**		
	Up to 1000	**1001 to 10000**	**Above 10000**
MedicalExpense	15	20	25
TravelExpense	10	15	20
FoodExpense	5	10	15

- Tax claim amount = Expense amount * (Tax percentage /100)
- Calculate the tax claim amount and display the same in result.jsp.

3.0 Component Specification

3.0.1 Controller

- The TaxController should be configured via annotation as a Controller

TaxController			
Attribute Name	**Attribute Type**	**Access Specifier**	**Constraints**
taxService	TaxServiceInterface	private	Use annotation to autowire

Method Name	**Method Argument name:type**	**Return type**	**RequestMapping URL & Request Method**
claimPage	modelAttribute “userClaim”:UserClaim	String	/getTaxClaimFormPage & GET
calculateTax	modelAttribute “userClaim”:UserClaim, result:BindingResult, map:ModelMap	String	/calculateTax & GET
populateExpense		List<String>	Should be annotated with ModelAttribute with name “expenseList”

- Inside the populateExpense method you should add all the 3-expense type into the list and return the list.
- TaxService interface should be autowired inside the Controller via annotations

- TaxServiceImpl must calculate the tax claim amount based on the Expense Entity that has the expenseType and the expenseAmount of the admin.

3.0.2 Service

- TaxService interface should be configured via annotation as Service

Method Name	**Method Argument name:type**	**Return type**	**Responsibilities**
calculateTax	“userClaim”:UserClaim	double	This method should calculate the claim amount based on the expense type and expense amount.

- TaxServiceImpl must implement TaxService interface.

3.0.3 Model

Class(Model)	**Property**	**Methods**
UserClaim	expenseType : String expenseAmt : double employeeId : String	//getters and setters

- Include the needed annotations inside the Expense Entity class for various validations already mentioned. The error messages should be retrieved from the properties file

3.0.4 Internationalization

- Override the methods in InternationalizationConfig class provided as part of the code skeleton to support Internationalization.

3.1 Request Mapping Tasks

/ getTaxClaimFormPage	Redirect the user to taxclaim.jsp
/calculateTax	Should invoke calculateTax method of the Controller after submitting all the valid expense details from taxclaim.jsp, which in turn should invoke the calculateTax method of the TaxServiceImpl and redirect to result.jsp with the appropriate values depicted in the screen design. If there are validation errors it should be redirected to taxclaim.jsp page

4.0 Overall Design Constraints

1) Implement the code using the best design standards.
2) Use Internationalization for all the labels and messages in the Tax Claim page.
3) Tax Claim page should support Internationalization for three languages English, German and French. The error messages and the labels should be displayed from the properties file depending on the locale. **The properties file is already provided as part of the code skeleton.**
4) Spring Validator should be used for all the Validations. Use only annotation for performing the validations.
5) TaxManagementApplication which is the main class, is already provided as part of the code skeleton. Include only the required annotations to auto scan the Controller, Service classes.
6) **Do not change the property name given in the application.properties files, you can change the value and you can include additional property if needed.**
7) **In the pom.xml you are provided with all the dependencies needed for developing the application.**
8) You will not be evaluated based on the UI design (layout, color, formatting, etc.). You are free to have a basic UI with all the required UI components (input fields, buttons, labels, etc.). Expected components with the id alone should be designed as per the requirement.
9) Adhere to the design specifications mentioned in the case study.
10) Do not change or delete the class/method names or return types which are provided to you as a part of the base code skeleton.
11) Also do change the name or id of the HTML component which are provided to you as a part of the base code skeleton.
12) **Please make sure that your code does not have any compilation errors while submitting your case study solution.**