

**Humana Encounter Resolution & Operation**

**Modify Time Card**

**Low Level Technical Design Document**

Prepared by: Prudhvi

Date: 03/14/19

Document Information

Revision History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Revision Number** | **Revision Date** | **Revision**  **By** | **Summary of Changes** | **Changes marked** |
| Initial | 03/14/2019 | Prudhvi | Initial Draft Version. | NA |
|  |  |  |  |  |

Approvals

This document requires following approvals.

|  |  |  |
| --- | --- | --- |
| Name | Title | Comments |
|  |  |  |
|  |  |  |

Purpose: Time card management screen contains all timecards information. A timecard can be added and updated by Admin user.

Table of Contents

1. [Timecard 4](#_Toc535307127)

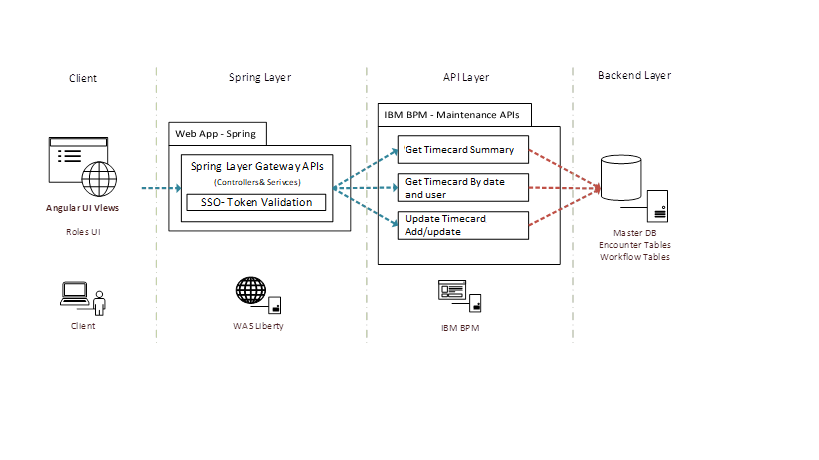
[I Angular View 5](#_Toc535307128)

II Validations……………………………………………………………………………………………………………………7 [III Spring API Layer](#_Toc535307129)……………………………………………………………………………………………………………8

*IV UI Layer………………………………………………………………………………………………………………………9*

## TimeCard

Maintain HERO application Timecard Management .



**Dependencies**

* UI
  + Interactive wireframe, HTML and CSS
  + Field matrix for field length, validations, default sort fields and etc.,
* Data
  + Tables
  + Sequences
  + Triggers for audit logs
  + Stored Procedures/Packages/Functions if required

**Services**

|  |  |  |  |
| --- | --- | --- | --- |
| **Service** | **Input** | **Output** | **Comments** |
| Get User Timecard Summary | Logged user Id  Search Criteria  Pagination Criteria | List of time card summary objects | This operation returns all time card summaries to display on GRID. |
| Get Lookup data (list of users) | Logged user Id  Type criteria | List of users based on type as user profile | This operation returns all user objects. |
| Get Lookup data (list of reason codes) | Logged user Id  Type criteria | List of timecard reason codes based on type as reason code | This operation returns all reason code objects. |
| Update timecard | Updated timecard object | Success/Failure  Error details | This operation saves a new timecard and also updates existing timecard if any changes. |
| Get timecard by date and user | Timecard object | List of all timecards for selected user | This operation returns all timecard details and record objects. |

**Sequence of Flow –**



1. **Angular View:** Based on user action(s) on view, Angular component controller identifies the service request and invoke Spring SPI service.

**Timecard Management screen flow:**

* 1. After clicking on Timecard Management link the request will transfer to features-routes based on configuration done in app.module.ts
  2. From features-routes the request will transfer to admin-routing.module.ts and find out Timecard Component.
  3. TimecardComponent contains associated timecardService. Here all common business service methods available.
  4. After that the request will caught by interceptor. ./src/app/core/shared/interceptor/index.ts is the interceptor file which contains interceptor details.
  5. Form index.ts it will go to DomainTransformerInterceptor.ts. DoaminTransformerInterceptor contains transformResponsePayload(response:any) method.
  6. DomainTransformerInterceptor contains transformResponsePayload(response:any) method.
  7. This method general identifies proper Domain Transformer so here it will identify TimecardDomainTransformer by using ConfigService. ConfigService contains all transformer’s information.
  8. ConfigService related configurations are available in
  9. TimecardDomainTransformer implements DomainTransformer. TimecardDomainTransformer contains transformResponsePayload(timecardSummary:Array<Object>)
  10. From here it will call create (config:any). From this method it will define all models. This whole process will be done at the time of loading application.

**To display all Timecards on Summary page:**

1. To display all timecards information, after select user from users dropdown and click button search it will call the method as getTimecardByUserId
2. Form here based on ConfigService it will find out appropriate Spring service called “TimecardSummaries” by using environment.ts configuration file and it uses appropriate TimecardSummaryDomainTransforer and display all summaries.

**To Add New Timecard:**

1. To add new Timecard we have to click Add New Timecard on beside of search button in screen. After clicking on add New Timecard the Add New Timecard popup will be displayed based on configuration done in html. The addTimecard () method will be called from timecard.component.ts. Here the default start date will be set after that Add timecard popup will be displayed.
2. After providing appropriate information we need to click on Save button. After clicking on Save button Save () method will be called. From here save service method will be called.
3. Form here based on ConfigService it will find out appropriate spring service called **“timecardSummary”** by using environment.ts configuration file and it uses appropriate TimecardSummaryDomainTransformer and display all summaries.

**To Update Existing Timecard:**

1. To update existing Timecard, first we need search from main screen of timecard. from here we select user from dropdown and give start date and end date (here start date and end date is optional) and click search button.
2. After clicking on search getTimeCardSummary() method will call and it also called configService and its domain transformer as getTimeCardSummary.domian transformer then we get results for timecard summery by selected user and display results in table format
3. Each row of table have edit button with icon will be there. Here we can click any edit button it will call getTimeCardByDate () method. In configService also call getTimeCardByDate. Domain transaformer
4. Form domain transformer we get results by using service as timecardByDate in environment.ts file. Once get the result from service should open update timecard popup and bind data from service we get.
5. If we can change any field or we can add any new timecard record then click on save call update () method and save updated fields or if any new fields are added.

**Lookup data:**

This lookup data will call from main screen onInit () time. In onInIt () method call getLookupData method in timecard service.

1. **Validations:**

* 1. For Timecard management validations are place in add timecard, update and timecard summary
  2. All Validation are done using NgForm library.
  3. In timecard summary screen without select user from dropdown validation message will show.
  4. In add timecard popup have html form. Here every field is mandatory so if any field was missing red color validation message will show at empty field place.
  5. The same validation was applicable for edit popup also.

**Start Date Validations:**

1. In timecard summary page start date is optional if end date giving without start date then validation message will show.
2. In Add time card popup by default start date is today date. Here we cannot select feature dates**,** only we can select previous or today date.
3. While update time start date is in disable mode.

**End Date Validations:**

In timecard summary page end date is optional. If start date and end date will give we check end date should not lesser than start date message will show at end date field place.

**Start Time and End Time Validations:**

In add timecard popup have start time and end time. Both fields are mandatory, and also check end time is not lesser than start time.

1. **Spring API Layer:** Based on the UI client layer request context, Spring Controller identifies the respective integration service and invokes the appropriate requested service.

**To fetch All Timecard Summaries by user:**

1. To fetch all timecard summaries by user, the Controller will call **public TimeCardSummaryResponse getAllTimeSummary (**
2. **@RequestParam(name = "\_paginate", required = false) Boolean paginate, @RequestParam(name = "\_page", required = false) Integer page,**

**@RequestParam (name = "\_limit", required = false) Integer size,**

**@RequestParam (name = "\_order", required = false) String order,**

**@RequestParam (name = "\_sort", required = false) String sort,**

**@RequestParam (name = "\_userId", required = true) String userId,**

**@RequestParam (name = "\_startDate", required = false) String startDate,**

**@RequestParam (name = "\_endDate", required = false) String endDate,**

Method with help of /v1/timecard/summaries. Here Pagination, Userd id, start date and end date related information will be added to request

1. From here public **public TimecardSummaryResponse getTimeCardSummary (TimeCardSummaryRequest request)** of TimecardService method will be called. From here **public TimeCardSummaryResponse getTimeCardSummary (TimeCardSummaryRequest request)** of TimecardDao method will be called.
2. The implementation of above method is available in TimecardBpmDao class. **Public TimecardSummaryResponse getTimeCardSummary(TimecardSummaryRequest request** method of TimecardBpmDao will be called.
3. From here with help of a **hero.timecard.gettimecardsummary.service.url** entry in application.properties the actual service will be called. It will fetch all timecard Summary details.
4. TimecardSummary request is the request and response is TimecardSummaryResponse.

**To Add Timecard:**

1. To add particular timecard, the Controller will call **public UpdateTimecardResponse (@RequestBody UpdateTimecardRequest request)** method with help of /v1/timecard.
2. From here **public UpdateTimecardResponse updateTimecard (UpdateTimecardRequest request)** of TimecardService method will be called.
3. From here **public UpdateTimecardResponse updateTimecard (UpdateTimecardRequest request)** of TimecardDao method will be called.
4. The implementation of above method is available in TimecardBpmDao class. **public UpdateTimecardResponse updateTimecard (UpdateTimecardRequest request)** method of TimecardBpmDao will be called.
5. From here with help of in **hero.timecard.updateUserTimecard.service.url** entry in application.dev.properties the actual service will be called after calling this service the record will be saved in Database.
6. UpdateTimecardRequest request is the request and response is updateTimecardResponse.

**To Update Timecard:**

1. To update Timecard the Controller will **call public UpdateTimecardResponse update (@RequestBody UpdateTimecardRequest request)** method with help of /v1/timecard.
2. From here **public UpdateTimecardResponse updateTimecard (UpdateTimecardRequest request)** of TimecardService method will be called.
3. From here **public UpdateTimecardResponse updateTimecard (UpdateTimecardRequest request)** of TimecardDao method will be called.
4. The implementation of above method is available in TimecardBpmDao class. **public UpdateTimecardResponse updateTimecard(UpdateTimecardRequest request)** method of TimecardBpmDao will be called.
5. From here with help of in application.dev.properties the actual service will be called after calling this service the record will be saved in Database.
6. UpdateTimecardRequest request is the request and response is updateTimecardResponse.

**Fetch Timecard By Date:**

1. To fetch timecard data by date controller will call public TimeCardDetailsByDateResponse getTimeCardByDate will call public TimeCardDetailsByDateResponse **getTimeCardByDate(@RequestBody TimecardDetailsByDateRequest request)**
2. From here **public** TimeCardDetailsByDateResponse **getTimeCardByDate ()** method will be called.
3. From here **public** TimeCardDetailsByDateResponse **getTimeCardByDate (TimecardDetailsByDateRequest request)** of TimecardBpmDao will be called.
4. From here with help of **hero.timecard.getTimecardByDate.service.url application.dev.properties** the actual service will be called.
5. TimecardDetailsByDateRequestis request and TimeCardDetailsByDateResponse is Response

### IV UI Layer

### UI components

|  |  |
| --- | --- |
| **Components** | **Implementation details** |
| View/Model | Timecard.component.html, TimecardSummaryModel.ts, TimecardSummary.ts |
| Component | Timecard.component.ts |
| Service | Timecard.service.ts |
| Spring Service Integration | Environment.ts |

#### Spring Components

Below table describes the high level UI Spring layers implementation classes’ details –

|  |  |
| --- | --- |
| **Components** | **Implementation details** |
| Controller | TimecardController |
| Service | TimecardService |
| DAO | TimecardDao is interface and TimecardBpmDao is implemented class |
| Configurations | Application.dev.properties |

* Spring **ConversionService** is being leverage to convert UI request object to service request object.
* Service request has additional parameter.

#### Data access objects (DAO’s)

Below table describes the backend service contracts details for the Current Case Dashboard chart.

|  |  |
| --- | --- |
| **Component** | **Details** |
| Rest Services | **To get all Timecard Summaries by user**   * [http://lxpbc004:9081/HEROUserManagmentServiceMedWeb/HEROUserManagmentServiceMedExport/getUserTimeCardsSummary](http://lxpbc004:9081/HEROUserManagmentServiceMedWeb/HEROUserManagmentServiceMedExport/getAllRoleSummary)   **To add/update Timecard:**   * [http://lxpbc004:9081/HEROUserManagmentServiceMedWeb/HEROUserManagmentServiceMedExport/updateUserTimecard](http://lxpbc004:9081/HEROUserManagmentServiceMedWeb/HEROUserManagmentServiceMedExport/updateRole)   **To get Timecard details by date**   * [http://lxpbc004:9081/HEROUserManagmentServiceMedWeb/HEROUserManagmentServiceMedExport/getUserTimecardByDate](http://lxpbc004:9081/HEROUserManagmentServiceMedWeb/HEROUserManagmentServiceMedExport/getRolePermissions)   **To get all users lookup service**   * [http://lxpbc004:9081/HEROUserManagmentServiceMedWeb/HEROUserManagmentServiceMedExport/getLookuData](http://lxpbc004:9081/HEROUserManagmentServiceMedWeb/HEROUserManagmentServiceMedExport/getAllPermissionsSummary) |