**TIME TRACKING – ESTIMATION FOR BACK END ARCHITECTURE MIGRATION**

* **Assumptions for the estimation:**
  + Assuming the API will have **already implemented (or at least mocked)** the following modules:
    - JWT authentication & RBAC to specific endpoints.
    - Connection to the MySQL DB.
    - Entities definition.
    - Basic REST endpoints to perform **CRUD** operations over the app entities/tables in the DB.
  + Assuming the main **business logic** (data handling & presentation) will still be on the **Java Spring back end**, which will serve the **JSP views** loading the data retrieved from the API to display on page load. After that, all the asynchronous calls and data requests from the front end will be directly against the API.
* **Estimation measurement:**
  + For each feature, the measures will be in the format ***Worst / Best / Estimated*** (approximately the **upper** **mean** between Best-Worst).
  + Specifying **N/A** when “Not Apply” or the feature is not affected by the migration.
  + Specific **clarifications** on the features will be marked with an **asterisk (\*).**
  + The **minimum** unit equivalence will be **1 Story Point = 1 working day (8h)**.

**JAVA BACK END SERVER – REST API CONNECTION**

**JWT authorization & RBAC from Back End** – 4 / 2 / 3

**Wrapper to perform API calls from Back End** – 4 / 2 / 3

**JSON responses handling & mapping to Java objects** **at Controller level\*** – 12 / 6 / 9

**(\*)** Counting time for every app’s entity mapping from JSON to Object, to be processed by the back-end business logic.

**LOGIN SCREEN**

**Login** (integration Active Directory with API JWT authorization after login) – 4 / 2 / 3

**Logout** and session timeout – N/A | The web session and cookie is destroyed by Spring, the timeout is set at server configuration and the timeout alert is handled only at front-end level.

**MAIL SENDING\*\***

**(\*)** All mail features may be maintained in the current system since the Java Mail with Spring is easy to configure and these features’ migration might **not be a high priority**.

**(\*)** If migrated, the first estimation below **includes work time to implement the new mail system** in the API.

**Monthly reminder** – 5 / 2 / 3

**Time Rejection** (Comments, Acceptance?) (external?) – 2 / 1 / 1

**Assign / Remove deputy** – 2 / 1 / 1

**FORCED REFRESH (FROM THE WEB PAGE)**

**Memory-Backed User Historic removal\*** – 15 / 8 / 12

(\*) To be decided if it will persist or not. This object is extremely coupled into every module and at every layer of the app’s back end (presentation, business, service…).

**USER INFO SCREEN**

**Get user data**

Get projects the user is member of the Team (All roles) – 2 / 1 / 1

Get own team users, their projects, and allocation (Managers) – 8 / 5 / 6

**Add people to my team** (Managers)

Find a person – 2 / 1 / 1

Select Projects/Add projects to existing member of the Team – 2 / 1 / 1

Allocation (date range and dedication percentage) – 2 / 1 / 1

**Remove user form my team** – 2 / 1 / 1

**Calculate input ratio** – 2 / 1 / 1

**Alert icon to pending-approval time of reporters** – 3 / 1 / 2

**Assign/Remove Deputy** – 3 / 1 / 2

**RECORD TIME**

**Get Data**

Select User (Managers) (first loading of users of their team) – 5 / 1 / 3

Select Project (first loading of available projects for each user) – 5 / 1 / 3

Select Week (Calendar Arrows or Week Picker) \* – 8 / 4 / 6

(\*) When the period changes, all registers for that period, user and their projects are requested.

**Calendar**

Copy & Paste – N/A | The week is backed up in memory at back-end level.

**Project handling**

Remove Project – N/A | The removal is performed at front-end level.

Add New Project – N/A | The addition is performed at front-end level.

**Time inputting**

Time totals calculation – N/A | The calculation is done at front-end level.

**Add Comments** – N/A | They are written in front end and saved with the time registers

**Save Registers** (new, deleted, approved/half-approved/to approve, …) – 12 / 8 / 10

**APPROVE TIME PER WEEK/MONTH**

**Get Data**

Select User (first loading of users of their team) – 5 / 1 / 3

Select Week (Calendar Arrows or Week Picker) \* – 8 / 4 / 6

(\*) When the period changes, all registers for that period, user and their projects are requested.

**Approve / Reject** – 4 / 1 / 2

**Enter Comments** – N/A | They are written in front end and saved with the time registers approved or rejected.

(\*) The estimations of these pages’ Get Data feature could be merged with their respective features in the Record Time page, some of whose data requests could have been already developed for that page. Because of this, the **overall Get Data feature estimation** maybe **the average of both pages’ estimations together**.

**MONTHLY SUMMARY\***

**Get Data**

Select Month (Arrows or Month Picker) \* – 8 / 4 / 6

(\*) When the period changes, all registers for that period, user and their projects are requested.

(\*) This estimation could be merged with Approve Time Per Month estimation, since the calendar is the same (monthly). In this case, the estimation **could only deviate around 2 story points up**.

**REPORTING \*\***

**(\*)** All reporting features may be maintained in the current system if the reports are not needed to be requested to the API from external apps/services, as these features’ migration cost may impact the estimation while **not being a high priority functional requirement** to be in the API.

**(\*)** If migrated, the first Report estimation below **includes work time to implement the new reporting structure & File generation** (Excel building) system in the API. Note that the other reports estimations **may decrease** as the previous have been implemented.

**Team Allocation Report** – 8 / 3 / 6

**Team Report** – 3 / 1 / 2

**Individual Report** – 3 / 1 / 2

**Personal Report** – 3 / 1 / 2

**All Users Report** – 3 / 1 / 2

**Total Allocation Report** – 3 / 1 / 2

**User List Report** – 3 / 1 / 2

**User Status Report** – 3 / 1 / 2

**Active Deputy Assignments Report** – 3 / 1 / 2

**Program/Service-Project Report** – 3 / 1 / 2

**ADMINISTRATION\*\***

(\*)Based on one of the assumptions first described in this document, if the basic CRUD operations were implemented/mocked for all app entities, the following estimations **count only the time to integrate the new API calls to each of these operations from the front-end**.

(\*) Note that the first estimation **includes the work time** that the others may see decreased by analogy to the first implementation.

**Create / Modify User** – 3 / 1 / 2

**Create / Modify Provider** – 2 / 1 / 1

**Create / Modify Program-Service** – 2 / 1 / 1

**Create / Modify Project** – 2 / 1 / 1

**Create / Modify Task** – 2 / 1 / 1

**Month Closure (Close / Open)** – 2 / 1 / 1

**DATABASE & SECURITY CHANGES**

**DB Schema refactorization**

PK, FK, table relationships – 4 / 2 / 3

Columns restrictions (type, size, UNIQUE/NOT NULL/DEFAULT values) – 3 / 1 / 2

**Spring Security configuration to apply RBAC at endpoint level** – 4 / 1 / 2