Design – EOD Process

[Packet approval / action/ Re-Register]

Background

The Supervisor logs in to the Registration Client and navigates to the end of day process page to view and act on ‘Pending Approval’, ‘Pending Action’ and ‘Re-register’ packets. Once the user select the packet and act on it then capture the authentication and save the information. Unauthenticated information should not be captured.

The **target users** are

* Supervisor

The key **requirements** are

Pending Approval:

* User views the list of Registration IDs pending approval.
* User selects a Registration ID to view registration details.
* User selects ‘Approve’. Alternatively, user selects ‘Reject’ or ‘On Hold’ along with the reason for rejection or holding.
* User repeats steps 3-4 for as many packets as desired.
* User clicks on ‘Authenticate’.
* User provides any one biometric - fingerprint, iris or face.
* User views confirmation of successful approval.

Pending Action:

* User views the list of Registration IDs which have been placed on hold.
* User selects a Registration ID to view registration details and the reason for placing on hold.
* User selects ‘Approve’. Alternatively, user selects ‘Reject’ along with the reason for rejection.
* User repeats steps 3-4 for as many packets as desired.
* User clicks on ‘Authenticate’.
* User provides any one biometric - fingerprint, iris or face.
* User views confirmation of successful approval.

Re-Register:

* User View the list of registration IDs that have been flagged as ‘re-register’ during packet status sync from the processor.
* On click of a registration ID, view the acknowledgement slip for the registration.
* Supervisor informs the individual by phone, email, physical mail or physical visit to re-register. This is an offline process.
  + Supervisor clicks on ‘Informed’.
* If unable to contact the **individual**, Supervisor clicks on ‘**Can’t inform**'.
  + On click of ‘Informed’ or ‘Can’t Inform' system shows the selected option. Retain the selected button in highlighted mode - to be confirmed with UX team. User can change the selection any number of times before authentication.
* Repeat steps 3-5 for one or more registration IDs in the ‘Re-register’ list.
* Click on ‘Authenticate’.
  + In the resulting pop-up, select the type of biometric the user is providing for authentication - fingerprint, iris or face. Further select the specific finger or iris being provided.
  + Scan the selected biometric.
  + Authenticate with locally stored biometric and display the result.
    - On successful authentication, the actioned packets are removed from the ‘-Re-register’ list.
    - On unsuccessful authentication, the user can retry his authentication with the same or a different biometric.

The key **non-functional requirements** are

* Security:
  + The bio data captured in the UI for authentication shouldn’t be stored in the system.
  + Once the validation completed then clear the same from memory.
* Log the each state of the packet view/approve/reject/hold:
  + As a security measures the UIN or any sensitive individual information should not be logged.
* Cache:
  + The bio data shouldn’t be cached into the session object.
* Other standard NFR, need to be taken care:
  + Logging, audit, exception handling.

Solution

The key solution considerations are –

* When user clicks the “EOD Process – Pending Approval / Action” option in menu tab, it should hit the controller [**RegistrationApprovalController.** getEnrollmentByStatus()] to fetch the list of registration packet pending for approval/ rejection / hold/ action process from database.
* **RegistrationApprovalController: -** for Pending Approval / Action
  + Invoke the interface **RegistrationApprovalService** and call the **getEnrollmentByStatus(String StatusCode)** to fetch list of registrations based on status code from db.
  + Display the fetched detail in the UI screen.
  + **userAuthenticate() –** when user clicks the submit button in the screen then display authentication.fxml file and initialize the respective FPAuthenticationController.java class and pass the current controller class as an argument to callback the process once authentication completed.
  + **updateRegistration(List<Registration>) –** Once the required registration ids are selected and clicked on the approval/ rejected/ onhold and authenticated by user then this method will be invoked and pass the required detail to respective service class.
* **FPAuthenticationController**: - for authentication
  + This controller should be called to authenticate the process done by the user in the screen by using his/ her finger print.
  + initialize (BaseController parentObj) – it initialize the controller and the respective biometric device. If device not available then display the respective error and close the screen.
  + capture() – when user clicks on ‘Scan’ button, this method should be invoked to capture the finger image and store the same in the respective object.
  + validate() – when user clicks on the ‘validate’ button in the pop-up window then validate the captured biometric against the value in the system [minutia based comparison]. If success then submit the page otherwise display the error message.
* **Re-RegistrationController:** – for re-registration
  + getAllReRegistrationPacket() – this method should invoke the Re-RegistrationService. getAllReRegistrationPacket() method to fetch and render the list of registrations in ‘Re-Register’ state.
  + updateReRegistrationStatus() – when user click on the submit button in the screen after authentication process this method should Re-RegistrationService.updateReRegistrationStatus() invoked to update the same in system.
* Create **RegistrationApprovalServiceImpl** and create DTO for the same.
  + **getEnrollmentByStatus(String statusCd)** – invokes the DAO class and fetch the client status code specific data from db and return the same to Controller class.
  + **updateRegistration(List<Registration>)** – it collects the information from controller and invoke the DAO class .
* **RegistrationApprovalDAOImpl**
  + **getEnrollmentByStatus(String statusCd)** – return all the registration detail based on the provided client status code.
  + **updateRegistration(List<Registration>)**  – this method updates the selected record status in the DB by calling the respective JPA repository interface.
* Send an Alert message (say) “Status updated successfully” or an error message.
* Handle exceptions in using custom Exception handler and send correct response to client.

**Classes**:

**UI**: RegistrationPendingApproval.fxml – for displaying of pending approval packets.

RegistrationPendingAction.fxml– for displaying of pending action packets in on-hold state.

Re-Registration.fxml – for displaying of registrations which are in ‘re-register’ state.

**Controller**:

RegistrationApprovalController.java,

Re-RegistrationController.java

FPAuthenticationController.java

**Service**: RegistrationApprovalServiceImpl.java, Re-RegistrationServiceImpl.java

**DAO**: RegistrationDAOImpl.java

**Repository:** RegistrationRepository.java

**DTO**: RegistrationApprovalDTO.java

Sequence and Class Diagram: for pending approval / action and Re-Registration case.

