Maadharapplication-algorithm

1. Start
2. Create a spring boot starter project maadharbackend for backend functionalities
3. Create a package com.maadhar.bean and create login and request entity classes in it
4. Declare the fields email(id) ,password and type of user in it
5. These will be mapped into column in databse
6. Request.java entity class is declared with fields aadharid,emailed,name,age,address
7. Aadhar id is developed as primary key in case of this class
8. Controller classes login controller and request controller is created
9. Login controller handles incoming http request related to userlogin and signup functions .
10. It receives jspn data for login and signup requests and convert to login object.
11. Request controller handles http request related to aadhar application request and deligate the processing to request service class
12. Requestadhar,getallreq,deletereq,approve,update,view methods are used with each method mapped to particular endpoint
13. Loginservice clss handles the logic for handling user login and signup functions.
14. Service class interact with login repository to access data.
15. Here sign in function checks the provided emailed exists in database using loginrepository.findbyid(login.getemailid()) method
16. Signup function handle signup function by checking email exists and if email exists,returns error message else follow signuppage
17. Requestservice class handles the request service for handling aadhar application request.here,
    - @Service: This annotation marks the class as a service component, indicating that it contains the business logic and should be automatically discovered by Spring.
    - @Autowired: This annotation enables Spring to automatically inject the RequestRepository bean into the service.
    - storeRequest: This method handles the submission of Aadhar Card application requests. It takes a Requests object representing the application details as input. The method calls the reqrepo.save(req) method to save the application details to the database and returns a success message
    - getAllRequests: This method retrieves all Aadhar Card application requests stored in the database. It uses the reqrepo.findAll() method to fetch the list of all requests.
    - denyRequest: This method handles the denial of a specific Aadhar Card application request. It takes the adharid variable as input, which represents the unique ID of the application to be denied. The method checks if the application with the provided adharid exists in the database using the reqrepo.findById(adharid) method. If the application exists, it deletes it from the database using the reqrepo.delete(p) method and returns a success message indicating that the application has been denied. Otherwise, it returns a message stating that the application is not present.
    - approveRequest: This method handles the approval of an Aadhar Card application request. It takes a Requests object representing the approved application details as input. The method checks if the application with the provided adharid exists in the database using the reqrepo.findById(req.getAdharid()) method. If the application exists, it updates the application details using the reqrepo.saveAndFlush(p) method and returns a success message indicating that the application has been approved. Otherwise, it returns a message stating that the application is not present.

- updateAdhar: This method handles the updation of Aadhar Card information. It takes a Requests object representing the updated application details as input. The method checks if the application with the provided adharid exists in the database using the reqrepo.findById(req.getAdharid()) method. If the application exists, it updates the application details (age, name, and address) and saves the changes to the database using the reqrepo.saveAndFlush(p) method. It returns a message indicating that the Aadhar Card updation has been requested. Otherwise, it returns a message stating that the Aadhar Card is not present.
- viewmyAdhar: This method retrieves the Aadhar Card details for a specific user based on their email ID. It takes the emailid variable as input, which represents the email ID of the user. The method calls the reqrepo.viewmyAdhar(emailid) method, which likely contains a custom query to fetch the Aadhar Card details for the specified user.

18. .for fronendapplication angular is used
19. Application is created with frontend working for login and signup fuctionalities
20. Admin and user can signup with respective credentials and can login to user and admin dashboard respectively
21. Admin dashboard provides view all requests and display all iisued aadhar card
22. Admin can approve or reject the request
23. Userdashboard have apply for new aadhar view option
24. Storereq contains new aadhar application data
25. And after successful approval of id user can view and update the aadharcard
26. After all these setupmethods,testing using testNG testclass is done
27. In loginservicetest,
    - WebDriver driver;: This line declares a WebDriver variable named "driver" to interact with the browser.
    - @BeforeTest: This annotation indicates that the method annotated with it will run before all the test methods in the class. In this case, the "config()" method is responsible for setting up the WebDriver and opening the browser.
    - WebDriverManager.chromedriver().setup();: This line sets up the ChromeDriver for WebDriver using the WebDriverManager library. It ensures that the appropriate ChromeDriver version is downloaded and configured automatically.
    - driver=new ChromeDriver();: This line creates a new instance of the ChromeDriver, which will control the Chrome browser.
    - driver.manage().window().maximize();: This line maximizes the browser window to ensure that the tests run in a maximized window.
    - @AfterTest: This annotation indicates that the method annotated with it will run after all the test methods in the class. In this case, the "afterClass()" method is responsible for closing the browser after the tests are executed.
    - driver.close();: This line closes the browser window after the tests are executed.
    - @Test: This annotation marks the methods that represent individual test cases.
    - signInTest(): This test case is intended to test the login functionality. It navigates the browser to the login page of the Aadhar Card application by calling driver.get("http://localhost:4200/login");.r

- signUpTest(): This test case is intended to test the signup functionality. It navigates the browser to the signup page of the Aadhar Card application by calling driver.get("http://localhost:4200/signup");

28. requestservice testclass is also declared this way and the application automation is completed
29. end