**Java FSD Capstone Project**

**mAaadhar Application**

**Algorithm:**

**Initialization:**

* Set up the required technologies and tools, such as MySQL for the database, Java (Spring Boot, JPA, Hibernate) for the backend, Angular, Bootstrap, HTML/CSS for the frontend, Selenium and TestNG for automation and testing, and Git, GitHub, Jenkins, and Docker for DevOps.

**Admin Portal:**

* Create an admin login page to authenticate admin credentials.
* Implement functionality to approve new Aadhaar Card requests.
* Implement functionality to verify requests for duplicate Aadhaar Cards.
* Display a list of all issued Aadhaar Cards.
* Enable the admin to delete Aadhaar Card details for deceased citizens.

**User Portal:**

* Design a user registration page for new citizens to sign up.
* Create a login page for users to access their accounts.
* Allow users to apply for a new Aadhar Card and submit necessary details.
* Allow users to view their assigned Aadhaar numbers after admin approval.
* Enable users to update their Aadhaar Card details, such as address, phone number, and date of birth.
* Provide an option for users to request a duplicate Aadhaar Card.

**Front-End Validation:**

* For admin passwords, perform validation to ensure they contain at least one uppercase letter, one lowercase letter, one special character, and one number.
* For citizen passwords, validate that they consist of only digits.

**Back-End Validation:**

* Validate the mobile number provided during registration.
* Ensure that the password length is not less than 6 characters.
* Verify the provided mobile number matches the mobile number in the Aadhaar database for citizen authentication.

**Backend Rest API:**

* Set up REST API endpoints for citizens' registration, admin approvals, duplicate Aadhaar verification, and other functionalities.
* Implement POST requests for citizens to register and apply for a new Aadhar Card.
* Implement POST requests for admin to approve Aadhaar Card requests and handle duplicate requests.

**Integration:**

* Integrate the backend APIs with the frontend to enable seamless communication between the user interface and backend services.

**Testing:**

* Use Selenium and TestNG to perform automated testing of different functionalities and user scenarios.

**Deployment and CI/CD:**

* Use Git, GitHub, Jenkins, and Docker for version control, continuous integration, and deployment of the application.

**User Interaction:**

* Allow users to interact with the application through the user portal for registration, application, and updates.
* Allow admins to interact with the application through the admin portal for approvals and verifications.

**Application Flow:**

* Implement the application flow to guide users through the registration, application, and verification processes efficiently.

**Error Handling:**

* Implement proper error handling and validation messages to provide meaningful feedback to users and admins.