Stable Image Core – Spring Boot + React Full Example

End-to-end sample that lets a React frontend send a prompt to a Spring Boot backend. The backend calls **Stability AI – Stable Image Core** via **OpenFeign** using `multipart/form-data`, then returns JSON with a base64 image to the frontend.

Project Structure

```
stable-image-demo/
├─ backend/
│  ├─ pom.xml
│  └─ src/main/java/com/example/stableimage/
│     ├─ StableImageApplication.java
│     ├─ config/FeignMultipartConfig.java
│     ├─ config/CorsConfig.java
│     ├─ client/StableImageClient.java
│     ├─ dto/GenerateRequest.java
│     ├─ dto/GenerateResponse.java
│     ├─ service/StableImageService.java
│     └─ web/StableImageController.java
│  └─ src/main/resources/application.yml
└─ frontend/
   ├─ package.json
   ├─ vite.config.js
   └─ src/
      ├─ main.jsx
      └─ App.jsx
```

---

1) Backend (Spring Boot)

`pom.xml`

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://
maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.example</groupId>
  <artifactId>stable-image-backend</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <properties>
```

```
    <java.version>17</java.version>
    <spring.boot.version>3.3.2</spring.boot.version>
    <spring.cloud.version>2023.0.3</spring.cloud.version>
    <feign.form.version>3.8.0</feign.form.version>
  </properties>

  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-dependencies</artifactId>
        <version>${spring.boot.version}</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
      <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-dependencies</artifactId>
        <version>${spring.cloud.version}</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
    </dependencies>
  </dependencyManagement>

  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <dependency>
      <groupId>org.springframework.cloud</groupId>
      <artifactId>spring-cloud-starter-openfeign</artifactId>
    </dependency>

    <!-- Feign form/multipart support -->
    <dependency>
      <groupId>io.github.openfeign.form</groupId>
      <artifactId>feign-form</artifactId>
      <version>${feign.form.version}</version>
    </dependency>
    <dependency>
      <groupId>io.github.openfeign.form</groupId>
      <artifactId>feign-form-spring</artifactId>
      <version>${feign.form.version}</version>
    </dependency>

    <dependency>
      <groupId>org.projectlombok</groupId>
      <artifactId>lombok</artifactId>
```

```
      <optional>true</optional>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-validation</artifactId>
    </dependency>
    <dependency>
      <groupId>com.fasterxml.jackson.core</groupId>
      <artifactId>jackson-databind</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-actuator</artifactId>
    </dependency>
  </dependencies>

  <build>
    <plugins>
      <plugin>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-maven-plugin</artifactId>
      </plugin>
    </plugins>
  </build>
</project>
```

`StableImageApplication.java`

```java
package com.example.stableimage;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.openfeign.EnableFeignClients;

@SpringBootApplication
@EnableFeignClients
public class StableImageApplication {
    public static void main(String[] args) {
        SpringApplication.run(StableImageApplication.class, args);
    }
}
```

`config/FeignMultipartConfig.java`

```java
package com.example.stableimage.config;

import feign.codec.Encoder;
import feign.form.spring.SpringFormEncoder;
import org.springframework.beans.factory.ObjectFactory;
```

```
import org.springframework.boot.autoconfigure.http.HttpMessageConverters;
import org.springframework.cloud.openfeign.support.SpringEncoder;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class FeignMultipartConfig {
    @Bean
    public Encoder feignFormEncoder(ObjectFactory<HttpMessageConverters>
messageConverters) {
        return new SpringFormEncoder(new SpringEncoder(messageConverters));
    }
}
```

config/CorsConfig.java

```
package com.example.stableimage.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.cors.CorsConfiguration;
import org.springframework.web.cors.UrlBasedCorsConfigurationSource;
import org.springframework.web.filter.CorsFilter;

import java.util.List;

@Configuration
public class CorsConfig {
    @Bean
    public CorsFilter corsFilter() {
        CorsConfiguration config = new CorsConfiguration();
        config.setAllowedOrigins(List.of("http://localhost:5173", "http://
localhost:3000"));

config.setAllowedMethods(List.of("GET","POST","PUT","DELETE","OPTIONS"));
        config.setAllowedHeaders(List.of("*"));
        config.setAllowCredentials(true);

        UrlBasedCorsConfigurationSource source = new
UrlBasedCorsConfigurationSource();
        source.registerCorsConfiguration("/**", config);
        return new CorsFilter(source);
    }
}
```

client/StableImageClient.java

```
package com.example.stableimage.client;
```

```java
import org.springframework.cloud.openfeign.FeignClient;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestHeader;
import org.springframework.web.bind.annotation.RequestPart;

import com.example.stableimage.config.FeignMultipartConfig;

import java.util.Map;

@FeignClient(
    name = "stableImageClient",
    url = "https://api.stability.ai/v2beta/stable-image/generate/core",
    configuration = FeignMultipartConfig.class
)
public interface StableImageClient {

    // JSON (base64) response
    @PostMapping(
        consumes = MediaType.MULTIPART_FORM_DATA_VALUE,
        headers = {"Accept=application/json"}
    )
    ResponseEntity<Map<String, Object>> generateJson(
            @RequestHeader("Authorization") String apiKey,
            @RequestPart("prompt") String prompt,
            @RequestPart(value = "aspect_ratio", required = false) String
aspectRatio,
            @RequestPart(value = "negative_prompt", required = false) String
negativePrompt,
            @RequestPart(value = "seed", required = false) Long seed,
            @RequestPart(value = "style_preset", required = false) String
stylePreset,
            @RequestPart(value = "output_format", required = false) String
outputFormat
    );

    // Raw image bytes response (optional convenience)
    @PostMapping(
        consumes = MediaType.MULTIPART_FORM_DATA_VALUE,
        headers = {"Accept=image/*"}
    )
    ResponseEntity<byte[]> generateBytes(
            @RequestHeader("Authorization") String apiKey,
            @RequestPart("prompt") String prompt,
            @RequestPart(value = "aspect_ratio", required = false) String
aspectRatio,
            @RequestPart(value = "negative_prompt", required = false) String
negativePrompt,
            @RequestPart(value = "seed", required = false) Long seed,
            @RequestPart(value = "style_preset", required = false) String
```

```
stylePreset,
            @RequestPart(value = "output_format", required = false) String
outputFormat
    );
}
```

dto/GenerateRequest.java

```
package com.example.stableimage.dto;

import jakarta.validation.constraints.NotBlank;
import lombok.Data;

@Data
public class GenerateRequest {
    @NotBlank
    private String prompt;

    // Optional fields
    private String aspectRatio;    // e.g., "1:1", "16:9"
    private String negativePrompt; // optional
    private Long seed;             // 0 or null = random
    private String stylePreset;    // e.g., "digital-art", "photographic"
    private String outputFormat;   // png | jpeg | webp (default png)
}
```

dto/GenerateResponse.java

```
package com.example.stableimage.dto;

import lombok.AllArgsConstructor;
import lombok.Data;

@Data
@AllArgsConstructor
public class GenerateResponse {
    private String format;  // png | jpeg | webp
    private String image;   // base64 data (no prefix)
    private Long seed;      // echo back used seed (if available)
}
```

service/StableImageService.java

```
package com.example.stableimage.service;

import com.example.stableimage.client.StableImageClient;
import com.example.stableimage.dto.GenerateRequest;
import com.example.stableimage.dto.GenerateResponse;
```

```java
import lombok.RequiredArgsConstructor;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Service;

import java.util.Base64;
import java.util.Map;

@Service
@RequiredArgsConstructor
public class StableImageService {

    private final StableImageClient client;

    @Value("${stability.api.key}")
    private String stabilityApiKey;

    public GenerateResponse generate(GenerateRequest req) {
        String format = req.getOutputFormat() != null ?
req.getOutputFormat() : "png";

        ResponseEntity<Map<String, Object>> response = client.generateJson(
                "Bearer " + stabilityApiKey,
                req.getPrompt(),
                req.getAspectRatio(),
                req.getNegativePrompt(),
                req.getSeed(),
                req.getStylePreset(),
                format
        );

        Map<String, Object> body = response.getBody();
        if (body == null) {
            throw new RuntimeException("Empty response from Stability API");
        }

        // The API returns base64 image in JSON. Typical key is "image".
        // Handle a few possible shapes defensively.
        String base64 = null;
        if (body.get("image") instanceof String s) {
            base64 = s;
        } else if (body.get("image_base64") instanceof String s2) {
            base64 = s2;
        } else if (body.get("images") instanceof Iterable<?> arr) {
            // if API returns an array, take the first
            for (Object v : arr) {
                if (v instanceof String vs) { base64 = vs; break; }
                if (v instanceof Map<?,?> m && m.get("image") instanceof
String vs2) { base64 = vs2; break; }
            }
        }
```

```
        if (base64 == null || base64.isBlank()) {
            throw new RuntimeException("Could not find base64 image in
Stability response: " + body);
        }

        // Validate base64
        try { Base64.getDecoder().decode(base64); } catch (Exception ex) {
            throw new RuntimeException("Invalid base64 image payload", ex);
        }

        Long usedSeed = req.getSeed() != null ? req.getSeed() : 0L;
        return new GenerateResponse(format, base64, usedSeed);
    }
}
```

web/StableImageController.java

```
package com.example.stableimage.web;

import com.example.stableimage.dto.GenerateRequest;
import com.example.stableimage.dto.GenerateResponse;
import com.example.stableimage.service.StableImageService;
import jakarta.validation.Valid;
import lombok.RequiredArgsConstructor;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/api/images")
@RequiredArgsConstructor
public class StableImageController {

    private final StableImageService service;

    @PostMapping("/generate")
    public ResponseEntity<GenerateResponse> generate(@Valid @RequestBody
GenerateRequest req) {
        return ResponseEntity.ok(service.generate(req));
    }
}
```

src/main/resources/application.yml

```
server:
  port: 8080

stability:
  api:
```

```
    key: ${STABILITY_API_KEY:CHANGE_ME}

logging:
  level:
    root: INFO
    com.example.stableimage: DEBUG
    org.springframework.cloud.openfeign: DEBUG
```

Run the backend

```
cd backend
export STABILITY_API_KEY=sk-****************************************
./mvnw spring-boot:run
```

(Windows PowerShell: `setx STABILITY_API_KEY your_key && mvnw spring-boot:run`)

---

2) Frontend (React + Vite)

`package.json`

```
{
  "name": "stable-image-frontend",
  "private": true,
  "version": "0.0.1",
  "type": "module",
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "preview": "vite preview"
  },
  "dependencies": {
    "axios": "^1.7.4",
    "react": "^18.2.0",
    "react-dom": "^18.2.0"
  },
  "devDependencies": {
    "@vitejs/plugin-react": "^4.3.1",
    "vite": "^5.4.0"
  }
}
```

`vite.config.js`

```
import { defineConfig } from 'vite'
import react from '@vitejs/plugin-react'
```

```
export default defineConfig({
  plugins: [react()],
  server: {
    port: 5173
  }
})
```

src/main.jsx

```
import React from 'react'
import ReactDOM from 'react-dom/client'
import App from './App'

ReactDOM.createRoot(document.getElementById('root')).render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
)
```

src/App.jsx

```
import { useState } from 'react'
import axios from 'axios'

const API_BASE = import.meta.env.VITE_API_BASE || 'http://localhost:8080'

export default function App() {
  const [prompt, setPrompt] =
useState('Lighthouse on a cliff overlooking the ocean')
  const [aspectRatio, setAspectRatio] = useState('1:1')
  const [stylePreset, setStylePreset] = useState('digital-art')
  const [outputFormat, setOutputFormat] = useState('png')
  const [negativePrompt, setNegativePrompt] = useState('')
  const [seed, setSeed] = useState('')

  const [imageUrl, setImageUrl] = useState('')
  const [loading, setLoading] = useState(false)
  const [error, setError] = useState('')

  const onGenerate = async () => {
    setLoading(true)
    setError('')
    setImageUrl('')
    try {
      const res = await axios.post(`${API_BASE}/api/images/generate`, {
        prompt,
        aspectRatio: aspectRatio || undefined,
        stylePreset: stylePreset || undefined,
        outputFormat: outputFormat || 'png',
```

```
        negativePrompt: negativePrompt || undefined,
        seed: seed ? Number(seed) : undefined
      })

      const { format, image } = res.data
      const dataUrl = `${image}`
      setImageUrl(dataUrl)
    } catch (e) {
      setError(e?.response?.data?.message || e.message ||
'Failed to generate image')
    } finally {
      setLoading(false)
    }
  }

  const canDownload = Boolean(imageUrl)

  return (
    <div style={{ maxWidth: 900, margin: '2rem auto', fontFamily: 'system-
ui' }}>
      <h1>Stable Image Core - Demo</h1>
      <p>Generate images via Spring Boot + Feign + Stability API</p>

      <div style={{ display: 'grid', gap: 12, gridTemplateColumns: '1fr
1fr' }}>
        <label>
          <div>Prompt</div>
          <textarea rows={4} value={prompt} onChange={e =>
setPrompt(e.target.value)} style={{ width: '100%' }} />
        </label>

        <div style={{ display: 'grid', gap: 12 }}>
          <label>
            <div>Aspect Ratio</div>
            <select value={aspectRatio} onChange={e =>
setAspectRatio(e.target.value)}>

{['1:1','16:9','9:16','3:2','2:3','4:5','5:4','21:9','9:21'].map(ar => (
                <option key={ar} value={ar}>{ar}</option>
              ))}
            </select>
          </label>

          <label>
            <div>Style Preset</div>
            <select value={stylePreset} onChange={e =>
setStylePreset(e.target.value)}>
              {['digital-art','photographic','cinematic','anime','comic-
book','fantasy-art','low-poly','pixel-art','line-art','3d-
model','isometric','analog-film','origami','neon-punk','tile-
texture','enhance','modeling-compound'].map(s => (
```

```
                <option key={s} value={s}>{s}</option>
              ))}
            </select>
          </label>

          <label>
            <div>Output Format</div>
            <select value={outputFormat} onChange={e =>
setOutputFormat(e.target.value)}>
              {['png','jpeg','webp'].map(f => (
                <option key={f} value={f}>{f}</option>
              ))}
            </select>
          </label>

          <label>
            <div>Negative Prompt (optional)</div>
            <input value={negativePrompt} onChange={e =>
setNegativePrompt(e.target.value)} />
          </label>

          <label>
            <div>Seed (optional)</div>
            <input type="number" value={seed} onChange={e =>
setSeed(e.target.value)} />
          </label>

          <button onClick={onGenerate} disabled={loading}>
            {loading ? 'Generating…' : 'Generate Image'}
          </button>
        </div>
      </div>

      {error && (
        <div style={{ marginTop: 16, color: 'crimson' }}>{error}</div>
      )}

      {imageUrl && (
        <div style={{ marginTop: 24 }}>
          <img src={imageUrl} alt="Generated" style={{ maxWidth: '100%',
borderRadius: 12, boxShadow: '0 4px 14px rgba(0,0,0,.15)' }} />
          <div style={{ marginTop: 8 }}>
            <a href={imageUrl} download={`stable-image.${outputFormat}`}
>Download</a>
          </div>
        </div>
      )}
    </div>
  )
}
```

Run the frontend

```
cd frontend
npm install
npm run dev
```

Open http://localhost:5173

You can set a custom backend URL via `.env`:

```
VITE_API_BASE=http://localhost:8080
```

How it works

1. **React** posts JSON to `/api/images/generate` with prompt + options.
2. **Spring Boot** converts that to a Feign `multipart/form-data` call (`@RequestPart`) and sets `Accept=application/json`.
3. **Stability API** returns JSON containing a **base64** image.
4. Backend returns `{ format, image, seed }` to the frontend.
5. React builds a `...` URL and displays it.

Troubleshooting

- **400: prompt required** → Feign not using multipart. Ensure `FeignMultipartConfig` bean and `consumes = MediaType.MULTIPART_FORM_DATA_VALUE`.
- **400: accept /** → Add `headers = {"Accept=application/json"}` in Feign method or pass `@RequestHeader("Accept")`.
- **403 moderation** → Adjust prompt and/or `negative_prompt`.
- **CORS** from browser → Update `CorsConfig` allowed origins to match your dev URL.
- **Different JSON key** → If Stability returns a key other than `image`, tweak the extraction in `StableImageService` accordingly.

Optional: Raw bytes endpoint

If you prefer direct image bytes instead of base64, add another controller method that calls `client.generateBytes(...)` and returns `image/*` with `ResponseEntity<byte[]>`.

```
// Example snippet inside controller
/*
@PostMapping(value = "/generate-bytes", produces = {"image/png","image/
jpeg","image/webp"})
public ResponseEntity<byte[]> generateBytes(@Valid @RequestBody
```

```
GenerateRequest req) {
    var resp = client.generateBytes(
        "Bearer " + stabilityApiKey,
        req.getPrompt(), req.getAspectRatio(), req.getNegativePrompt(),
        req.getSeed(), req.getStylePreset(), req.getOutputFormat()
    );
    return ResponseEntity.status(resp.getStatusCode()).body(resp.getBody());
}
*/
```

That's it ✅

- Put your Stability API key in env var `STABILITY_API_KEY`.
- Start backend (`mvn spring-boot:run`) and frontend (`npm run dev`).
- Generate images from the UI.

If you want, I can tailor the UI styles, add history, seeds, or a gallery grid next.