



VIT[®]
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

JAVA PROJECT

EduSmart – AI Powered Educational Tool

By:
Akshita Mathur - 25MCA0066

1. Introduction

EduSmart is a Java-based, AI-powered educational web application developed to enhance learning efficiency by automating the summarization and quiz generation processes. The project leverages the Gemini AI API for content processing and integrates it seamlessly with Java's backend capabilities. The aim is to simplify the learning process by converting long, complex notes into concise summaries and automatically generated quizzes.

EduSmart combines artificial intelligence with traditional programming techniques, enabling the creation of an intelligent educational assistant. The project demonstrates strong backend design using Java, emphasizing networking, file handling, and real-time API interaction to produce a practical, user-friendly solution for students.

2. Objectives

The objective of EduSmart is to create a platform that intelligently processes academic material using AI while demonstrating advanced Java concepts. The project's design ensures that it serves as both an academic tool and a technical demonstration of how modern AI can integrate with Java applications.

- Automate note simplification using Gemini AI.
- Generate quizzes directly from uploaded educational content.
- Implement REST-style backend communication using Java sockets.
- Enable users to download simplified notes as text files.
- Showcase API communication, multithreading, and JSON parsing in Java.

3. Technology Stack

EduSmart utilizes a hybrid stack where the frontend provides a simple user interface, and the backend, built with pure Java, handles all the logic and AI interaction. The technology stack ensures performance, scalability, and modularity.

- **Frontend:** HTML, CSS, and JavaScript – For building an intuitive, interactive user interface.
- **Backend:** Core Java – Using ServerSocket, File I/O, and HttpURLConnection for API communication.
- **Libraries and APIs:** Gson – For converting and parsing JSON responses from Gemini API.
- **Gemini API** – For intelligent text simplification and quiz generation.
- **Development Environment:** Visual Studio Code / Eclipse IDE.

4. Key Features

EduSmart includes a variety of features that demonstrate Java's flexibility and integration capabilities with external AI systems. Each feature has been implemented keeping both functionality and code clarity in mind.

- AI-driven text simplification for concise understanding of notes.
- Automatic quiz generation for learning reinforcement.
- Multithreaded backend to handle multiple client requests concurrently.
- Secure file upload and download feature for user data management.
- Real-time API communication using Java's networking features.
- User-friendly interface with quick response feedback.

5. System Design and Functionality

EduSmart follows a client-server architecture where the frontend acts as the client and the Java backend as the server. When the user uploads a file and chooses an action (Simplify Notes or Generate Quiz), the frontend sends an HTTP request to the Java backend. The backend processes this request, sends a structured prompt to the Gemini API, and retrieves AI-generated content.

The backend server is implemented using Java's ServerSocket and HttpURLConnection classes. Each client request is handled by a separate thread through handler classes such as SimplifyHandler and QuizHandler, ensuring concurrent performance. The server then parses the Gemini API response using Gson and returns it as plain text to the frontend, which the user can view or download as a .txt file.

6. Implementation and Screenshots

6.1 User Interface Screenshots

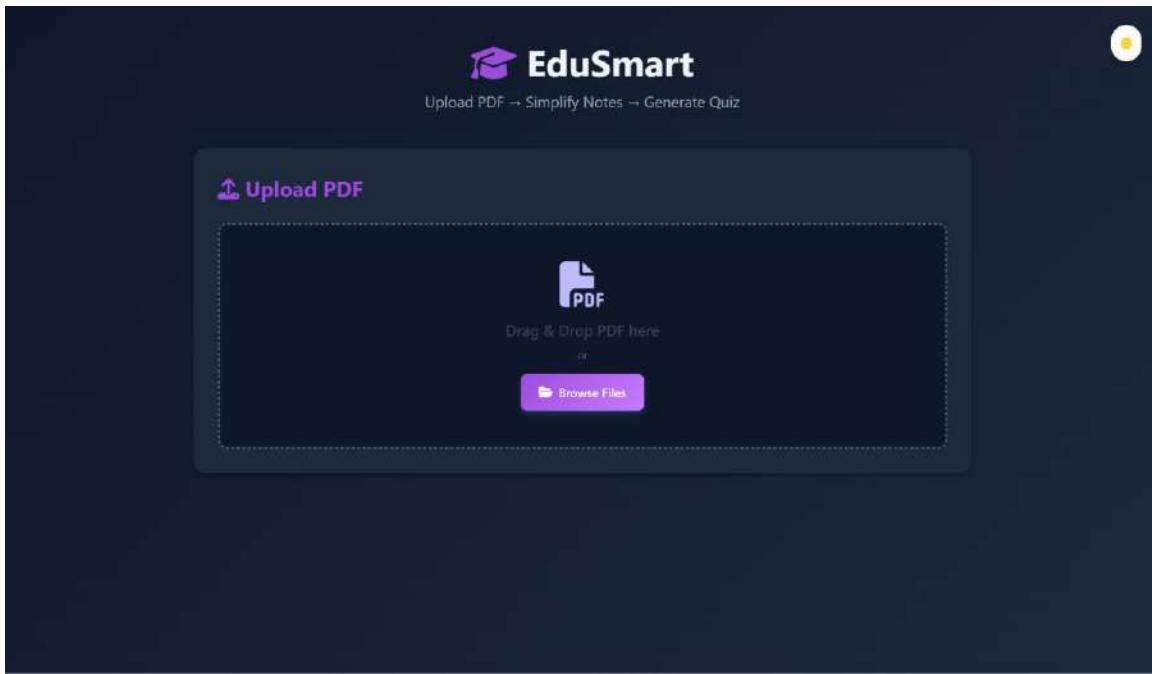


Fig 6.1.1 Dark Theme

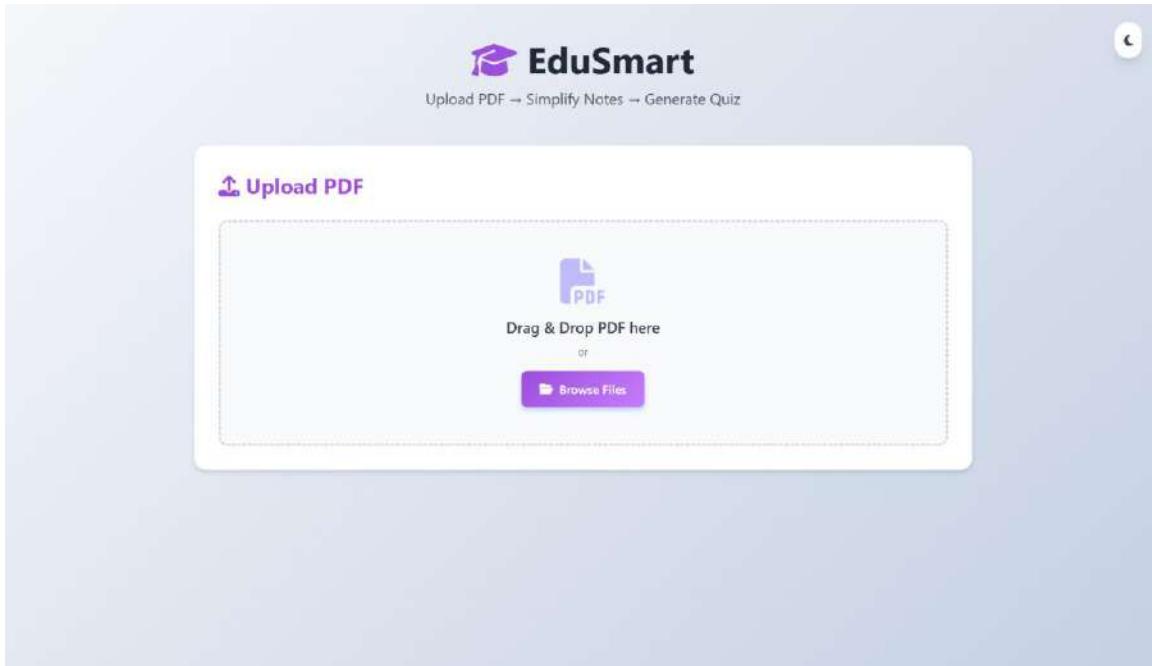


Fig 6.1.2 Light Theme

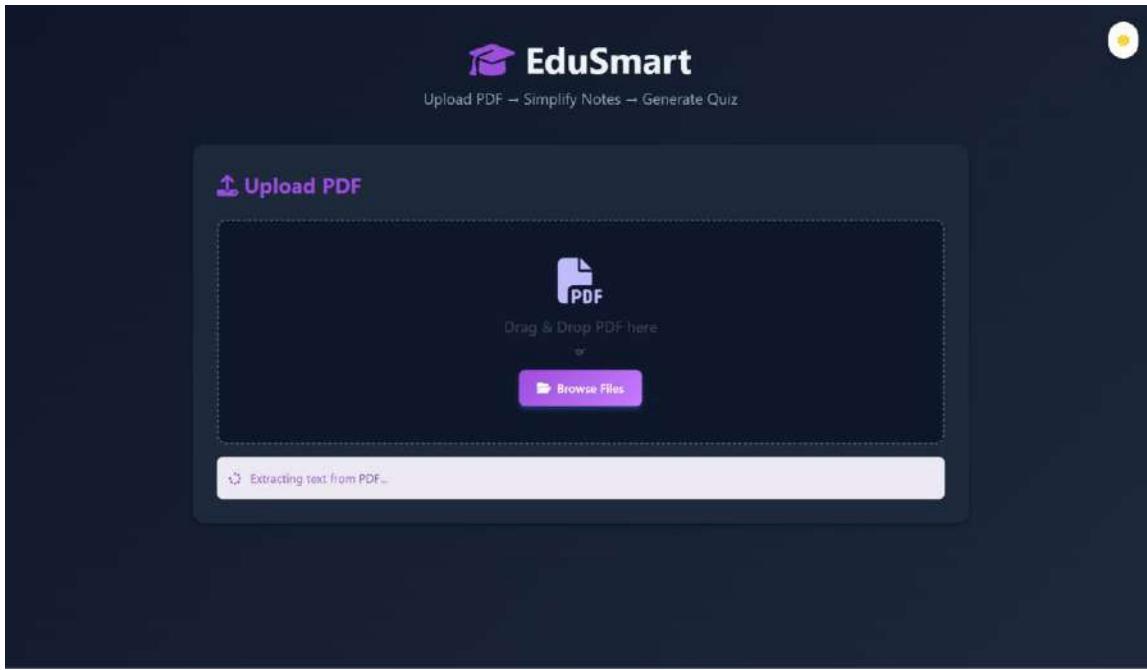


Fig 6.1.3 Extraction + Simplification of Uploaded Text

A screenshot of the EduSmart website's simplified notes page. The title "Simplified Notes" is displayed above a list of numbered points. Points 3 through 8 describe MapReduce concepts like Key-Value Pairs, Fault Tolerance, Data Locality, HDFS storage, architecture, and replication. Below this is a "SUMMARY" section with a detailed explanation of MapReduce. At the bottom are "Download Notes" and "Generate Quiz" buttons.

Simplified Notes

The MapReduce Programming Paradigm
MapReduce is a special way of programming designed to process very large amounts of data.
* **Scalability:** It can easily expand to use hundreds or thousands of computers, each with many processors.
* **Massive Data:** It handles "web-scale" data, which means hundreds of gigabytes, terabytes, or even petabytes (1,000 terabytes).
* **Distributed Storage:** Because the data is so huge, it often won't fit on a single computer. So, it needs a special distributed file system, like Google File System (GFS) or Hadoop Distributed File System (HDFS), to store data across many machines.

Why MapReduce is Needed (Motivations)
MapReduce was created for several key reasons:
* **Large-scale Data Processing:** To handle very big datasets on groups of computers (clusters).
* **Massive Parallelism:** To run tasks simultaneously on hundreds or thousands of CPUs.
* **Reliability:** To ensure tasks complete successfully, even if some computers fail, and to make data easy to access.
* **Key Functions:**
 * **Automatic Parallelization & Distribution:** It automatically splits tasks and sends them to different computers.
 * **Fault-tolerance:** It can recover from failures (e.g., if a computer stops working).
 * **Monitoring Tools:** It provides tools to check the status of tasks.
 * **Clean Abstraction:** It offers a simple way for programmers to write code without worrying about the complex parallel processing details.
* **It combines "functional programming" ideas with "distributed computing" and it maintains the "batch processing".

[Download Notes](#) [Generate Quiz](#)

6.1.4 Downloadable Simplified Notes

Quiz Time!

Question 1 of 8

What fundamental problem is MapReduce designed to solve efficiently?

Real-time transaction processing for financial systems.

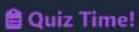
Processing extremely large datasets across a cluster of computers.

Storing data in a traditional relational database management system.

Managing small, frequently updated files on a single server.

[Next Question →](#)

Fig 6.1.5 Generated Quiz Questions

 Quiz Time!



Score: 5/8 (63%)

Q1: What is the fundamental problem MapReduce was designed to solve?
Correct answer: Efficiently processing massive datasets by distributing computations across a cluster of commodity computers.

Q2: During the MapReduce process, what is the primary function of the 'Map' phase?
Correct answer: To process individual chunks of input data independently and emit intermediate key-value pairs.

Q3: According to the HDFS block placement policy, where are the second and third replicas of a data block typically placed?
Correct answer: The second replica is placed on a different rack, and the third replica is placed on the "same rack" as the second replica.

Q4: In the Hadoop MapReduce framework, what is the main responsibility of the Partitioner?
Correct answer: To decide which Reducer will process each intermediate key-value pair.

Q5: What is the primary mechanism MapReduce employs to achieve fault tolerance for individual tasks?
Correct answer: Automatically restarting failed tasks on other healthy nodes.

Q6: Why does the HDFS NameNode store all of its critical metadata in main memory?
Correct answer: To enable extremely fast access and updates to the file system namespace.

Q3: According to the HDFS block placement policy, where are the second and third replicas of a data block typically placed?
Correct answer: The second replica is placed on a different rack, and the third replica is placed on the "same rack" as the second replica.

Q4: In the Hadoop MapReduce framework, what is the main responsibility of the Partitioner?
Correct answer: To decide which Reducer will process each intermediate key-value pair.

Q5: What is the primary mechanism MapReduce employs to achieve fault tolerance for individual tasks?
Correct answer: Automatically restarting failed tasks on other healthy nodes.

Q6: Why does the HDFS NameNode store all of its critical metadata in main memory?
Correct answer: To enable extremely fast access and updates to the file system namespace.

Q7: How does the Hadoop MapReduce Job Tracker prioritize scheduling for Map tasks compared to Reduce tasks regarding data locality?
Correct answer: Map tasks are preferably scheduled on nodes close to their input data, whereas Reduce tasks are assigned to any available Task Tracker.

Q8: What is the primary purpose of speculative execution in MapReduce?
Correct answer: To run an extra copy of a slow-running task (straggler) on a different machine to finish the job faster.



Fig 6.1.6 Final Score of the Generated Quiz

6.2 Code Implementation

6.2.1 EduSmartServer.java

```
7. import com.sun.net.httpserver.HttpServer;
8. import com.sun.net.httpserver.HttpHandler;
9. import com.sun.net.httpserver.HttpExchange;
10. import java.io.*;
11. import java.net.InetSocketAddress;
12. import java.nio.file.Files;
13. import java.nio.file.Paths;
14.
15. public class EduSmartServer {
16.     private static final int PORT = 8080;
17.
18.     public static void main(String[] args) throws IOException {
19.         HttpServer server = HttpServer.create(new
InetSocketAddress(PORT), 0);
20.
21.         // Routes
22.         server.createContext("/", new FileHandler());
23.         server.createContext("/upload", new UploadHandler());
24.         server.createContext("/simplify", new SimplifyHandler());
25.         server.createContext("/quiz", new QuizHandler());
26.
27.         server.setExecutor(null);
28.         server.start();
29.
30.         System.out.println("EduSmart Server started at
http://localhost:" + PORT);
31.         System.out.println("Open your browser and visit:
http://localhost:8080");
32.     }
33.
34.     static class FileHandler implements HttpHandler {
35.         @Override
36.         public void handle(HttpExchange exchange) throws IOException {
37.             String path = exchange.getRequestURI().getPath();
38.             if (path.equals("/")) path = "/index.html";
39.
40.             File file = new File("web" + path);
41.
42.             if (file.exists() && !file.isDirectory()) {
```

```
43.             String contentType = getContentType(path);
44.             exchange.getResponseHeaders().set("Content-Type",
45.                 contentType);
46.
47.             OutputStream os = exchange.getResponseBody();
48.             Files.copy(file.toPath(), os);
49.             os.close();
50.         } else {
51.             String response = "404 - File Not Found";
52.             exchange.sendResponseHeaders(404, response.length());
53.             OutputStream os = exchange.getResponseBody();
54.             os.write(response.getBytes());
55.             os.close();
56.         }
57.     }
58.
59.     private String getContentType(String path) {
60.         if (path.endsWith(".html")) return "text/html";
61.         if (path.endsWith(".css")) return "text/css";
62.         if (path.endsWith(".js")) return "application/javascript";
63.         return "text/plain";
64.     }
65. }
66.
67. // Handle PDF upload
68. static class UploadHandler implements HttpHandler {
69.     @Override
70.     public void handle(HttpExchange exchange) throws IOException {
71.         if ("POST".equals(exchange.getRequestMethod())) {
72.             InputStream is = exchange.getRequestBody();
73.             ByteArrayOutputStream buffer = new
    ByteArrayOutputStream();
74.
75.             byte[] data = new byte[1024];
76.             int bytesRead;
77.             while ((bytesRead = is.read(data)) != -1) {
78.                 buffer.write(data, 0, bytesRead);
79.             }
80.
81.             String filename = "uploads/temp_" +
    System.currentTimeMillis() + ".pdf";
82.             Files.write(Paths.get(filename), buffer.toByteArray());
83.         }
84.     }
85. }
```

```
84.             String extractedText =
85.                 PDFProcessor.extractText(filename);
86.
87.             String jsonResponse = "{\"text\": " +
88.                 escapeJson(extractedText) + "}";
89.
90.             exchange.getResponseHeaders().set("Content-Type",
91.                 "application/json");
92.             exchange.sendResponseHeaders(200,
93.                 jsonResponse.getBytes().length);
94.             OutputStream os = exchange.getResponseBody();
95.             os.write(jsonResponse.getBytes());
96.             os.close();
97.
98.
99.         // Simplify notes
100.        static class SimplifyHandler implements HttpHandler {
101.            @Override
102.            public void handle(HttpExchange exchange) throws IOException
103.            {
104.                if ("POST".equals(exchange.getRequestMethod())) {
105.                    String text = new
106.                        String(exchange.getRequestBody().readAllBytes());
107.
108.                    String simplified = NoteSimplifier.simplify(text);
109.
110.                    String jsonResponse = "{\"simplified\": " +
111.                        escapeJson(simplified) + "}";
112.
113.                    exchange.getResponseHeaders().set("Content-Type",
114.                        "application/json");
115.                    exchange.sendResponseHeaders(200,
116.                        jsonResponse.getBytes().length);
117.                    OutputStream os = exchange.getResponseBody();
118.                    os.write(jsonResponse.getBytes());
119.                    os.close();
120.                }
121.            }
122.
123.        }
124.
125.    }
126.
127.    // Generate quiz
```

```

120.     static class QuizHandler implements HttpHandler {
121.         @Override
122.         public void handle(HttpExchange exchange) throws IOException
123.         {
124.             if ("POST".equals(exchange.getRequestMethod()))
125.             {
126.                 String text = new
127.                     String(exchange.getRequestBody().readAllBytes());
128.
129.                 String quiz = QuizGenerator.generate(text);
130.
131.                 exchange.getResponseHeaders().set("Content-Type",
132.                     "application/json");
133.                 exchange.sendResponseHeaders(200,
134.                     quiz.getBytes().length);
135.                 OutputStream os = exchange.getResponseBody();
136.                 os.write(quiz.getBytes());
137.                 os.close();
138.             }
139.         }
140.     }
141.
142.     private static String escapeJson(String text) {
143.         return "\"" + text.replace("\\\"", "\\\\"")
144.             .replace("\\"", "\\\"");
145.             .replace("\n", "\\n")
146.             .replace("\r", "\\r")
147.             .replace("\t", "\\t") + "\"";
148.     }
149. }
```

6.2.2 NoteSimplifier.java

```

7  import com.google.gson.JsonObject;
8  import com.google.gson.JsonParser;
9  import com.google.gson.JsonArray;
10 import java.net.URI;
11 import java.net.http.HttpClient;
12 import java.net.http.HttpRequest;
13 import java.net.http.HttpResponse;
14
15 public class NoteSimplifier {
16
```

```
17     private static final String API_KEY =
18         "AIzaSyAiyMk4TrbXzFApn1o0uapgoxE2TzOHu1E";
19     private static final String API_URL =
20         "https://generativelanguage.googleapis.com/v1beta/models/gemini-2.5-
21         flash:generateContent";
22
23     public static String simplify(String rawText) {
24         rawText = rawText.trim();
25
26
27
28
29         try {
30             System.out.println("Calling Gemini API for note
31 simplification...");
32
33             String prompt = createSmartPrompt(rawText);
34
35             String response = callGeminiAPI(prompt);
36
37             System.out.println("Gemini API call successful!");
38             return response;
39
40         } catch (Exception e) {
41             System.err.println("ERROR calling Gemini API: " +
42             e.getMessage());
43             e.printStackTrace();
44             return "ERROR: Failed to simplify notes using AI.\n\n" +
45                 "Error message: " + e.getMessage() + "\n\n" +
46                 "Please check:\n" +
47                 "1. Your API key is correct\n" +
48                 "2. You have internet connection\n" +
49                 "3. Gemini API is accessible in your region";
50
51     }
52
53     private static String createSmartPrompt(String text) {
54
55         if (text.length() > 15000) {
56             text = text.substring(0, 15000) + "...";
57         }
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
```

```
57         return "You are an expert educator helping students learn.  
58         Your task is to simplify this educational content into easy-to-  
59         understand notes.\n\n" +  
60  
61         "IMPORTANT INSTRUCTIONS:\n" +  
62         "- Write in simple, clear language that a student can  
63         easily understand\n" +  
64         "- Break down complex concepts into simple  
65         explanations\n" +  
66         "- Use examples and analogies where helpful\n" +  
67         "- Keep the original order and flow of topics\n" +  
68         "- Be comprehensive - don't skip important  
69         information\n" +  
70         "- Only include sections that are actually present in  
71         the content\n" +  
72         "- If there are no definitions/examples/terms in the  
73         content, skip those sections entirely\n\n" +  
74  
75         "FORMAT YOUR RESPONSE LIKE THIS:\n\n" +  
76  
77         "\u263a INTRODUCTION\n" +  
78         "[Write 2-3 sentences introducing what this content is  
79         about]\n\n" +  
80  
81         "\u263a MAIN CONTENT\n" +  
82         "[Break the content into logical sections. For each  
83         major topic/concept:\n" +  
84         "- Give it a clear heading\n" +  
85         "- Explain it in simple terms\n" +  
86         "- Use bullet points for lists\n" +  
87         "- Add examples if helpful]\n\n" +  
88  
89         "\u263a KEY TAKEAWAYS\n" +  
90         "[List the 5-8 most important points students should  
91         remember]\n\n" +  
92  
93         "\u263a SUMMARY\n" +  
94         "[Write a comprehensive 3-4 sentence summary of  
95         everything covered]\n\n" +  
96  
97         "NOW SIMPLIFY THIS CONTENT:\n\n" + text;  
98     }  
99  
100    private static String callGeminiAPI(String prompt) throws  
101        Exception {
```

```
90     HttpClient client = HttpClient.newHttpClient();
91
92     //JSON payload for Gemini
93     JsonObject content = new JsonObject();
94     JsonArray contents = new JsonArray();
95     JsonObject message = new JsonObject();
96     JsonArray parts = new JsonArray();
97     JsonObject part = new JsonObject();
98
99     part.addProperty("text", prompt);
100    parts.add(part);
101    message.add("parts", parts);
102    contents.add(message);
103    content.add("contents", contents);
104
105    String jsonPayload = content.toString();
106
107    String urlWithKey = API_URL + "?key=" + API_KEY;
108
109    //HTTP request
110    HttpRequest request = HttpRequest.newBuilder()
111        .uri(URI.create(urlWithKey))
112        .header("Content-Type", "application/json")
113        .POST(HttpRequest.BodyPublishers.ofString(jsonPayload))
114        .build();
115
116    // Send request
117    HttpResponse<String> response = client.send(request,
118        HttpResponse.BodyHandlers.ofString());
119
120    System.out.println("API Response Status: " +
121        response.statusCode());
122
123    if (response.statusCode() != 200) {
124        System.err.println("API Response Body: " +
125            response.body());
126        throw new Exception("API returned status code: " +
127            response.statusCode() + ". Check your API key and quota.");
128    }
129
130    JsonObject jsonResponse =
131        JsonParser.parseString(response.body()).getAsJsonObject();
132
133    if (jsonResponse.has("candidates")) {
```

```

130         JSONArray candidates =
131             jsonResponse.getAsJSONArray("candidates");
132             if (candidates.size() > 0) {
133                 JSONObject candidate =
134                     candidates.get(0).getAsJsonObject();
135                     JSONObject contentObj =
136                         candidate.getAsJsonObject("content");
137                     JSONArray partsArray =
138                         contentObj.getAsJSONArray("parts");
139                         if (partsArray.size() > 0) {
140                             return
141                                 partsArray.get(0).getAsJsonObject().get("text").getAsString();
142                             }
143                         }
144                     }
145                 }
146             }
147             throw new Exception("Unexpected API response format");
148         }
149     }

```

6.2.3 QuizGenerator.java

```

7  import com.google.gson.Gson;
8  import com.google.gson.JsonObject;
9  import com.google.gson.JsonParser;
10 import com.google.gson.JsonArray;
11 import java.net.URI;
12 import java.net.http.HttpClient;
13 import java.net.http.HttpRequest;
14 import java.net.http.HttpResponse;
15
16 public class QuizGenerator {
17
18     private static final String API_KEY =
19         "AIzaSyAiyMk4TrbXzFApn1o0uapgoxE2TzOHu1E";
20     private static final String API_URL =
21         "https://generativelanguage.googleapis.com/v1beta/models/gemini-2.5-
22         flash:generateContent";
23
24     public String generateQuiz() {
25         String response = HttpClient.newHttpClient()
26             .send(HttpRequest.newBuilder()
27                 .uri(URI.create(API_URL))
28                 .header("Content-Type", "application/json")
29                 .header("Authorization", "Bearer " + API_KEY)
30                 .method(HttpMethod.POST, HttpRequest.BodyPublishers.ofString(
31                     "{"prompt": "What is the capital of France?"}")));
32         return response;
33     }
34 }

```

```
21     public static String generate(String text) {
22         text = text.trim();
23
24         if (text.isEmpty()) {
25             return "{\"questions\": []}";
26         }
27
28
29         try {
30             System.out.println("Calling Gemini API for quiz
generation...");
31
32             String prompt = createQuizPrompt(text);
33
34             String response = callGeminiAPI(prompt);
35
36             String validatedJson = extractAndValidateJSON(response);
37
38             System.out.println("Gemini API call successful! Quiz
generated.");
39             return validatedJson;
40
41         } catch (Exception e) {
42             System.err.println("ERROR calling Gemini API: " +
e.getMessage());
43             e.printStackTrace();
44             return createErrorQuiz();
45         }
46     }
47
48     private static String createQuizPrompt(String text) {
49         if (text.length() > 15000) {
50             text = text.substring(0, 15000) + "...";
51         }
52
53         return "You are an expert educator creating a quiz to test
student understanding.\n\n" +
54
55             "TASK: Create exactly 8 multiple-choice questions
based on the content below.\n\n" +
56
57             "REQUIREMENTS FOR EACH QUESTION:\n" +
58             "1. Test actual understanding of the content (not
trivial facts)\n" +
59             "2. Have 4 options (only ONE is correct)\n" +
```

```

60             "3. Wrong options should be plausible but clearly
incorrect\n" +
61             "4. Questions should cover different parts of the
content\n" +
62             "5. Mix of difficulty levels (some easy, some
challenging)\n" +
63             "6. Clear, unambiguous wording\n\n" +
64
65             "CRITICAL: You MUST respond with ONLY valid JSON in
this EXACT format:\n\n" +
66             "{\n" +
67                 "\"questions\": [\n" +
68                     "{\n" +
69                         "\"question\": \"Your question here?\",\n" +
70                         "\"options\": [\"Option A\", \"Option B\",
\"Option C\", \"Option D\"],\n" +
71                             "\"correctAnswer\": 0\n" +
72                         },\n" +
73                         "... 7 more questions ...\n" +
74                     ]\n" +
75                 }\n\n" +
76
77             "Where correctAnswer is the index (0, 1, 2, or 3) of
the correct option.\n" +
78             "Do NOT include any text before or after the JSON.\n"
+
79             "Do NOT use markdown code blocks.\n" +
80             "Just pure JSON.\n\n" +
81
82             "CONTENT TO CREATE QUIZ FROM:\n\n" + text;
83     }
84
85     private static String callGeminiAPI(String prompt) throws
Exception {
86         HttpClient client = HttpClient.newHttpClient();
87
88         JsonObject content = new JsonObject();
89         JSONArray contents = new JSONArray();
90         JsonObject message = new JsonObject();
91         JSONArray parts = new JSONArray();
92         JsonObject part = new JsonObject();
93
94         part.addProperty("text", prompt);
95         parts.add(part);
96         message.add("parts", parts);

```

```
97         contents.add(message);
98         content.add("contents", contents);
99
100        String jsonPayload = content.toString();
101
102        String urlWithKey = API_URL + "?key=" + API_KEY;
103
104        HttpRequest request = HttpRequest.newBuilder()
105            .uri(URI.create(urlWithKey))
106            .header("Content-Type", "application/json")
107            .POST(HttpRequest.BodyPublishers.ofString(jsonPayload))
108            .build();
109
110        HttpResponse<String> response = client.send(request,
111            HttpResponse.BodyHandlers.ofString());
112
113        System.out.println("API Response Status: " +
114            response.statusCode());
115
116        if (response.statusCode() != 200) {
117            System.err.println("API Response Body: " +
118                response.body());
119            throw new Exception("API returned status code: " +
120                response.statusCode());
121
122        JsonObject jsonResponse =
123            JsonParser.parseString(response.body()).getAsJsonObject();
124
125        if (jsonResponse.has("candidates")) {
126            JSONArray candidates =
127                jsonResponse.getAsJSONArray("candidates");
128            if (candidates.size() > 0) {
129                JsonObject candidate =
130                    candidates.get(0).getAsJsonObject();
131                JsonObject contentObj =
132                    candidate.getAsJsonObject("content");
133                JSONArray partsArray =
134                    contentObj.getAsJSONArray("parts");
135                if (partsArray.size() > 0) {
136                    return
137                        partsArray.get(0).getAsJsonObject().get("text").getAsString();
138                }
139            }
140        }
141    }
142}
```

```
133
134     if (jsonResponse.has("error")) {
135         JSONObject error = jsonResponse.getAsJsonObject("error");
136         String errorMsg = error.get("message").getAsString();
137         throw new Exception("Gemini API Error: " + errorMsg);
138     }
139
140     throw new Exception("Unexpected API response format");
141 }
142
143     private static String extractAndValidateJSON(String response) {
144         response = response.replaceAll(`\`json\\s*`,
145                                         "").replaceAll(``\s*`, "").trim();
146
147         int startIndex = response.indexOf("{");
148         int endIndex = response.lastIndexOf("}");
149
150         if (startIndex != -1 && endIndex != -1) {
151             response = response.substring(startIndex, endIndex + 1);
152         }
153
154         try {
155             JSONObject quiz =
156             JsonParser.parseString(response).getAsJsonObject();
157
158             if (!quiz.has("questions")) {
159                 throw new Exception("Missing 'questions' field");
160             }
161
162             JSONArray questions = quiz.getAsJSONArray("questions");
163             for (int i = 0; i < questions.size(); i++) {
164                 JSONObject q = questions.get(i).getAsJsonObject();
165
166                 if (!q.has("question") || !q.has("options") ||
167                     !q.has("correctAnswer")) {
168                     throw new Exception("Invalid question structure
169                     at index " + i);
170                 }
171
172                 JSONArray options = q.getAsJSONArray("options");
173                 if (options.size() != 4) {
174                     throw new Exception("Question " + i + " must have
175                     exactly 4 options");
176                 }
177             }
178         }
179     }
```

```
173             int correctAnswer =
174                 q.get("correctAnswer").getAsInt();
175                 if (correctAnswer < 0 || correctAnswer > 3) {
176                     throw new Exception("Invalid correctAnswer at
177                         question " + i);
178                 }
179             return response;
180
181         } catch (Exception e) {
182             System.out.println("JSON validation error: " +
e.getMessage());
183             System.out.println("Response was: " + response);
184             return createErrorQuiz();
185         }
186     }
187
188     private static String createErrorQuiz() {
189         JSONObject result = new JSONObject();
190         JSONArray questions = new JSONArray();
191
192         JSONObject q1 = new JSONObject();
193         q1.addProperty("question", "The quiz generator requires AI
API configuration. What should you do?");
194         JSONArray options1 = new JSONArray();
195         options1.add("Get a FREE Gemini API key from
https://aistudio.google.com/app/apikey");
196         options1.add("Give up on this feature");
197         options1.add("Use a different app");
198         options1.add("Hope it fixes itself");
199         q1.add("options", options1);
200         q1.addProperty("correctAnswer", 0);
201         questions.add(q1);
202
203         JSONObject q2 = new JSONObject();
204         q2.addProperty("question", "After getting the API key, what's
the next step?");
205         JSONArray options2 = new JSONArray();
206         options2.add("Paste it in QuizGenerator.java where it says
YOUR_GEMINI_API_KEY_HERE");
207         options2.add("Email it to someone");
208         options2.add("Post it on social media");
209         options2.add("Keep it secret and never use it");
210         q2.add("options", options2);
```

```

211         q2.addProperty("correctAnswer", 0);
212         questions.add(q2);
213
214         result.add("questions", questions);
215
216         Gson gson = new Gson();
217         return gson.toJson(result);
218     }
219 }
```

6.2.4 App.js

```

7      let extractedText = '';
8      let simplifiedNotes = '';
9      let quizData = null;
10     let userAnswers = {};
11
12     const themeToggle = document.getElementById('themeToggle');
13     const htmlElement = document.documentElement;
14     const moonIcon = themeToggle.querySelector('.fa-moon');
15     const sunIcon = themeToggle.querySelector('.fa-sun');
16
17     function applyTheme(theme) {
18         if (theme === 'dark') {
19             htmlElement.classList.add('dark');
20
21             moonIcon.classList.add('hidden');
22             sunIcon.classList.remove('hidden');
23         } else {
24             htmlElement.classList.remove('dark');
25
26             moonIcon.classList.remove('hidden');
27             sunIcon.classList.add('hidden');
28         }
29         localStorage.setItem('theme', theme);
30     }
31
32     const currentTheme = localStorage.getItem('theme') || 'light';
33     applyTheme(currentTheme);
34
35     themeToggle.addEventListener('click', () => {
36
37         const newTheme = htmlElement.classList.contains('dark') ?
'light' : 'dark';

```

```
38         applyTheme(newTheme);
39     });
40
41
42     // PDF Upload
43     const uploadArea = document.getElementById('uploadArea');
44     const pdfInput = document.getElementById('pdfInput');
45     const uploadStatus = document.getElementById('uploadStatus');
46     const notesSection = document.getElementById('notesSection');
47     const notesContent = document.getElementById('notesContent');
48
49     // browse
50     uploadArea.addEventListener('click', () => {
51         pdfInput.click();
52     });
53
54     pdfInput.addEventListener('change', (e) => {
55         const file = e.target.files[0];
56         if (file && file.type === 'application/pdf') {
57             uploadPDF(file);
58         } else {
59             alert('Please upload a valid PDF file');
60         }
61     });
62
63     // Drag and drop
64     uploadArea.addEventListener('dragover', (e) => {
65         e.preventDefault();
66         uploadArea.classList.add('border-blue-500', 'bg-blue-50');
67     });
68
69     uploadArea.addEventListener('dragleave', (e) => {
70         e.preventDefault();
71         uploadArea.classList.remove('border-blue-500', 'bg-blue-50');
72     });
73
74     uploadArea.addEventListener('drop', (e) => {
75         e.preventDefault();
76         uploadArea.classList.remove('border-blue-500', 'bg-blue-50');
77
78         const file = e.dataTransfer.files[0];
79         if (file && file.type === 'application/pdf') {
80             uploadPDF(file);
81         } else {
82             alert('Please upload a valid PDF file');
```

```
83         }
84     });
85
86     // Upload PDF to server
87     async function uploadPDF(file) {
88         uploadStatus.classList.remove('hidden');
89
90         try {
91             const formData = new FormData();
92             formData.append('pdf', file);
93
94             const response = await fetch('/upload', {
95                 method: 'POST',
96                 body: file
97             });
98
99             if (!response.ok) {
100                 throw new Error('Upload failed');
101             }
102
103             const data = await response.json();
104             extractedText = data.text;
105
106             await simplifyNotes(extractedText);
107
108         } catch (error) {
109             console.error('Error:', error);
110             alert('Error uploading PDF. Please try again.');
111             uploadStatus.classList.add('hidden');
112         }
113     }
114
115     // Simplify Notes
116     async function simplifyNotes(text) {
117         uploadStatus.innerHTML = '<i class="fas fa-spinner fa-spin mr-2"></i> Simplifying notes...';
118
119         try {
120             const response = await fetch('/simplify', {
121                 method: 'POST',
122                 headers: {
123                     'Content-Type': 'text/plain'
124                 },
125                 body: text
126             });

```

```
127      if (!response.ok) {
128          throw new Error('Simplification failed');
129      }
130
131      const data = await response.json();
132      simplifiedNotes = data.simplified;
133
134      // Display
135      notesContent.innerHTML = formatNotes(simplifiedNotes);
136      notesSection.classList.remove('hidden');
137      uploadStatus.classList.add('hidden');
138
139      notesSection.scrollIntoView({ behavior: 'smooth' });
140
141  } catch (error) {
142      console.error('Error:', error);
143      alert('Error simplifying notes. Please try again.');
144      uploadStatus.classList.add('hidden');
145  }
146
147 }
148
149 function formatNotes(text) {
150
151     let formatted = text.replace(/\n/g, '<br>');
152
153     formatted = formatted.replace(/KEY DEFINITIONS/g, '<strong
154         class="text-2xl text-blue-600 dark:text-blue-400">KEY
155         DEFINITIONS</strong>');
156     formatted = formatted.replace(/KEY POINTS/g, '<strong
157         class="text-2xl text-green-600 dark:text-green-400">KEY
158         POINTS</strong>');
159     formatted = formatted.replace(/SUMMARY/g, '<strong
160         class="text-2xl text-purple-600 dark:text-purple-400">SUMMARY</strong>');
161
162     formatted = formatted.replace(/\• /g, '<span class="text-blue-
163         500">•</span> ');
164
165     formatted = formatted.replace(/\={50}/g, '<span class="text-
166         gray-300 dark:text-gray-
167         600">=====
```

```
163 // Download
164 const downloadBtn = document.getElementById('downloadBtn');
165
166 downloadBtn.addEventListener('click', () => {
167     const content = notesContent.innerText;
168
169     const blob = new Blob([content], { type: 'text/plain' });
170     const url = URL.createObjectURL(blob);
171     const a = document.createElement('a');
172     a.href = url;
173     a.download = 'simplified_notes.txt';
174     a.click();
175     URL.revokeObjectURL(url);
176 });
177
178
179 // Generate Quiz
180 const generateQuizBtn = document.getElementById('generateQuizBtn');
181 const quizSection = document.getElementById('quizSection');
182 const quizContainer = document.getElementById('quizContainer');
183 const nextQuestionBtn = document.getElementById('nextQuestionBtn');
184 const quizResults = document.getElementById('quizResults');
185 const restartBtn = document.getElementById('restartBtn');
186
187 let currentQuestion = 0;
188 generateQuizBtn.addEventListener('click', async () => {
189     generateQuizBtn.innerHTML = '<i class="fas fa-spinner fa-spin mr-2"></i> Generating Quiz...';
190     generateQuizBtn.disabled = true;
191
192     try {
193         const response = await fetch('/quiz', {
194             method: 'POST',
195             headers: { 'Content-Type': 'text/plain' },
196             body: simplifiedNotes
197         });
198
199         if (!response.ok) throw new Error('Quiz generation failed');
200
201         const data = await response.json();
202         quizData = data.questions;
203
204         currentQuestion = 0;
205         userAnswers = {};
206     }
```

```
207     quizSection.classList.remove('hidden');
208     nextQuestionBtn.classList.add('hidden');
209     quizResults.classList.add('hidden');
210     restartBtn.classList.add('hidden');
211
212     showQuestion();
213
214     generateQuizBtn.innerHTML = '<i class="fas fa-question-circle mr-2"></i> Generate Quiz';
215     generateQuizBtn.disabled = false;
216     quizSection.scrollIntoView({ behavior: 'smooth' });
217
218 } catch (error) {
219     alert('Error generating quiz. Try again.');
220     generateQuizBtn.innerHTML = '<i class="fas fa-question-circle mr-2"></i> Generate Quiz';
221     generateQuizBtn.disabled = false;
222 }
223 });
224
225 // current question
226 function showQuestion() {
227     const q = quizData[currentQuestion];
228
229     quizContainer.innerHTML =
230         <h2 class="text-2xl font-bold mb-6">
231             Question ${currentQuestion + 1} of ${quizData.length}
232         </h2>
233         <p class="text-lg mb-6">
234             ${q.question}
235         </p>
236         <div class="space-y-4">
237             ${q.options.map((opt, i) =>
238                 <button id="opt${i}" class="answer-option w-full p-4 rounded-lg border bg-white hover:bg-purple-100 text-gray-900 transition-all text-left"
239                     onclick="selectAnswer(${i})">${opt}</button>
240             ).join('')}
241         </div>
242     ;
243
244     nextQuestionBtn.classList.add('hidden');
245 }
246
247 window.selectAnswer = function(optionIndex) {
```

```
248     userAnswers[currentQuestion] = optionIndex;
249
250     document.querySelectorAll(".answer-option").forEach(btn =>
251       btn.classList.remove("selected"));
252
253     document.getElementById("opt" + optionIndex).classList.add("selected");
254
255   };
256
257 // Next question button
258 nextQuestionBtn.addEventListener('click', () => {
259   currentQuestion++;
260
261   if (currentQuestion < quizData.length) {
262     showQuestion();
263     nextQuestionBtn.classList.add('hidden');
264   } else {
265     showResults();
266   }
267 });
268
269 // Results
270 function showResults() {
271   let correct = 0;
272   quizData.forEach((q, index) => {
273     if (userAnswers[index] === q.correctAnswer) correct++;
274   });
275
276   const total = quizData.length;
277   const percentage = Math.round((correct / total) * 100);
278
279   let resultHTML =
280     `

### Score: ${correct}/${total} (${percentage}%)


281     <div class="mt-6 space-y-4 text-left">
282     `;
283
284   quizData.forEach((q, i) => {
285     const isCorrect = userAnswers[i] === q.correctAnswer;
286     const correctText = q.options[q.correctAnswer];
287
288     resultHTML += `
```

```

289         <div class="p-4 rounded-lg ${isCorrect ? 'bg-green-100
290           dark:bg-green-900/30' : 'bg-red-100 dark:bg-red-900/30'}">
291           <p class="font-semibold">Q${i+1}: ${q.question}</p>
292           <p class="">Correct answer: <span class="font-
293             bold">${correctText}</span></p>
294         </div>`;
295     });
296
297     resultHTML += `</div>`;
298
299     quizContainer.innerHTML = '';
300     quizResults.innerHTML = resultHTML;
301     quizResults.classList.remove('hidden');
302     nextQuestionBtn.classList.add('hidden');
303     restartBtn.classList.remove('hidden');
304   }
305
306   // Restart
307   restartBtn.addEventListener('click', () => {
308     location.reload();
309   });
310
311   console.log('EduSmart app.js loaded successfully!');

```

6.2.5 index.html

```

7  <!DOCTYPE html>
8  <html lang="en">
9    <head>
10      <meta charset="UTF-8">
11      <meta name="viewport" content="width=device-width, initial-
12        scale=1.0">
13      <title>EduSmart - Smart Note Simplifier</title>
14
15      <script src="https://cdn.tailwindcss.com"></script>
16      <link rel="stylesheet" href="styles.css">
17      <link rel="stylesheet"
18        href="https://cdnjs.cloudflare.com/ajax/libs/font-
19          awesome/6.4.0/css/all.min.css">
20    </head>
21    <body class="bg-gray-50 dark:bg-gray-900 transition-colors duration-
22        300">
23
24      <!-- Toggle -->

```

```
21      <button id="themeToggle" class="fixed top-6 right-6 z-50 p-3  
22        rounded-full bg-white dark:bg-gray-800 shadow-lg hover:shadow-xl  
23        transition-all">  
24        <i class="fas fa-moon dark:hidden text-gray-700"></i>  
25        <i class="fas fa-sun hidden dark:inline text-yellow-400"></i>  
26      </button>  
27  
28      <div class="min-h-screen py-12 px-4">  
29        <div class="max-w-5xl mx-auto">  
30          <header class="text-center mb-12">  
31            <h1 class="text-5xl font-bold text-gray-800  
dark:text-white mb-3">  
32              <i class="fas fa-graduation-cap text-[#9d4edd]"></i> EduSmart  
33            </h1>  
34            <p class="text-xl text-gray-600 dark:text-gray-300">  
35              Upload PDF → Simplify Notes → Generate Quiz  
36            </p>  
37          </header>  
38  
39          <!-- Upload Section -->  
40          <section id="uploadSection" class="card mb-8">  
41            <h2 class="section-title">  
42              <i class="fas fa-upload"></i> Upload PDF  
43            </h2>  
44  
45            <div class="upload-area" id="uploadArea">  
46              <i class="fas fa-file-pdf text-6xl text-[#bebcfc]  
mb-4"></i>  
47              <p class="text-xl font-semibold text-gray-700  
dark:text-gray-300 mb-2">  
48                Drag & Drop PDF here  
49              </p>  
50              <p class="text-sm text-gray-500 dark:text-gray-  
400 mb-4">or</p>  
51              <button class="btn-primary">  
52                <i class="fas fa-folder-open mr-2"></i>  
53                Browse Files  
54              </button>  
55              <input type="file" id="pdfInput" accept=".pdf"  
hidden>  
56            </div>
```

```
56             <div id="uploadStatus" class="hidden mt-4 p-4  
57                 rounded-lg bg-purple-50 dark:bg-blue-900/30 text-[#9d4edd] dark:text-[#9d4edd]">  
58                 <i class="fas fa-spinner fa-spin mr-2"></i>  
59             Extracting text from PDF...  
60         </div>  
61     </section>  
62  
63     <!-- Simplified Notes -->  
64     <section id="notesSection" class="card mb-8 hidden">  
65         <h2 class="section-title">  
66             <i class="fas fa-file-alt"></i> Simplified Notes  
67         </h2>  
68  
69         <div id="notesContent" contenteditable="true"  
70             class="notes-editor">  
71             </div>  
72  
73         <div class="flex gap-4 mt-6">  
74             <button id="downloadBtn" class="btn-primary">  
75                 <i class="fas fa-download mr-2"></i> Download  
76                 Notes  
77             </button>  
78             <button id="generateQuizBtn" class="btn-  
79                 secondary">  
80                 <i class="fas fa-question-circle mr-2"></i>  
81                 Generate Quiz  
82             </button>  
83         </div>  
84     </section>  
85  
86     <!-- Quiz Section -->  
87     <section id="quizSection" class="card hidden">  
88         <h2 class="section-title">  
89             <i class="fas fa-clipboard-list"></i> Quiz Time!  
90         </h2>  
91  
92         <div id="quizContainer" class="text-left p-4 rounded-lg"></div>  
93  
94         <button id="nextQuestionBtn" class="btn-primary mt-6 hidden">  
95             Next Question →  
96         </button>
```

```

93      <div id="quizResults" class="hidden mt-6 p-6 rounded-lg bg-
  [#ede9fe] dark:bg-green-900/30"></div>
94
95      <button id="restartBtn" class="btn-secondary mt-6 hidden">
96          Upload New PDF
97      </button>
98  </section>
99
100     </div>
101 </div>
102
103     <footer class="text-center py-8 text-gray-600 dark:text-gray-
  400">
104
105     </footer>
106
107     <script src="app.js"></script>
108 </body>
109 </html>

```

7. Project Demonstration

- Video Link:

[https://drive.google.com/file/d/1ZSuNIp67HAQa7AQwiQMZG0zFjyNL_OtA/view?
usp=sharing](https://drive.google.com/file/d/1ZSuNIp67HAQa7AQwiQMZG0zFjyNL_OtA/view?usp=sharing)

- Code:

[https://drive.google.com/drive/folders/1gQh8vEQkinbftCYlvXE3hi8h_zvCCdmC?us
p=sharing](https://drive.google.com/drive/folders/1gQh8vEQkinbftCYlvXE3hi8h_zvCCdmC?usp=sharing)

8. Conclusion

EduSmart demonstrates the power of combining Core Java with modern artificial intelligence capabilities. The project successfully integrates Gemini API with a Java backend, utilizing multithreading, file handling, and network programming. It showcases not only technical expertise but also practical innovation in applying AI to educational contexts.

This project reflects the adaptability of Java in integrating with cutting-edge AI tools and demonstrates how traditional programming languages can still be at the forefront of intelligent software solutions. EduSmart is a testament to the potential of combining logic-driven backend systems with AI to enhance learning experiences.