

**COLLEGE CERTIFICATE**

This is to certify that this is the bonafide record of the application development entitled,”An Adaptive Social Media Recommendation system” Submitted by Y.Vishnu Vardhan(2211CS020553),Y.Prudhvi Charan(2211 B. Tech II year I semester, Department of CSE (AI&ML) during the year 2023-24. The results embodied in the report have not been submitted to any other university or institute for the award of any degree or diploma

## External Examiner

|  |  |  |
| --- | --- | --- |
|  |  |  |

## ACKNOWLEDGEMENT

We have been truly blessed to have a wonderful internal guide **Mrs. SUCHITRA PATTABIRAMAN, Asst .Prof, Department of CSE(AIML)**, **Malla Reddy University** for guiding us to explore the ramification of our work and we express our sincere gratitude towards him for leading me through the completion of Project.

We would like to say our sincere thanks to **Mr. k. Vikram, Asst.Professor, Department of CSE(AIML)**, App Development Coordinator, for providing seamless support and right suggestions are given in the development of the APP.

We would like to say our sincere thanks to **Mrs. Lakshmi. T.K, Incharge & Assistant Professor, Department of CSE I. B.Tech, Mallareddy University** for providing seamless support and right suggestions are given in the development of the APP.

We would like to say our sincere thanks to **Mr. J. Shashi Kumar, Incharge &**

**Asst.Prof, Department of AIML,I B.Tech, Mallareddy University** for providing

Seamless support and right suggestions are given in the development of the APP.

We wish to express our sincere thanks to **Dr. V. Dhanunjana Chari, Dean SOS & I B. Tech SOE, Mallareddy University** for providing us with the conducive environment for carrying through our academic schedules and Project with ease.

We wish to express our sincere thanks to **Vice Chancellor sir** and **The Management** of **Mallareddy University** for providing excellent infrastructure and their visionary thoughts to prepare ourselves industry ready by focusing on new technologies.

Finally, we would like to thank our family members and friends for their moral support and encouragement to achieve goals.

### P. JAHNAVI (2211CS020557)

### P. RAJINIKANTH (2211CS020559)

### P. LIKITHA (2211CS020562)

### P. MOHITH (2211CS020620)

### P. NIKHIL REDDY(2211CS020621)

## ABSTRACT

Java and Python are two of the most popular and powerful programming languages of present time. Both of them are Object-Oriented programming languages with unique advantages for developers and end users.In this app, we develop JPT, which is a translator that converts Java code into Python. Our desktop application takes Java code as an input and translates it to Python code using XML as an intermediate language.The translator enables this conversion instead of having to rewrite the whole Python program from start.Providing an open-source translator which discloses conversion steps from source to intermediate to target language enables academics and professionals to gain more insight on how to best modify code such that it is error free after conversion.

This work will contribute in the possible switch from Java to Python by helping reduce the software evolution cost as well as help Java programmers to learn Python.The Java to Python Converter not only supports the conversion of individual code files but also facilitates the translation of entire Java projects into Python. It can handle complex dependencies, automatically converting imported Java packages to their corresponding Python modules and ensuring proper integration.

# TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO.** | **TITLE** | **PAGE NO.** |
| 1 | **Introduction to APP**  **1.1 Summary of application**  **1.2 Background of application** | 1 |
| 2 | **Requirements of application** | 5 |
| 3 | **Design- algorithm or flow chart**  **3.1 Screen Shots of APP** | 6 |
| 4 | **Writing your App's Code**  **4.1 Handling Errors** | 11 |
| 5 | **Conclusion** | 25 |
| 6 | **Future scope** | 26 |

## CHAPTER 1

## INTRODUCTION TO APP

The Java to Python Converter is a tool designed to assist in converting Java code into equivalent Python code. It aims to automate the process of converting Java programs to Python, helping developers who are familiar with Java transition to Python programming more easily.

Python and Java are both popular programming languages, but they have different syntax, structures, and programming paradigms. As a result, manually converting Java code to Python can be a time-consuming and error-prone task. The Java to Python Converter automates this process, saving developers valuable time and effort

Syntax Conversion: The converter analyzes the Java code and translates its syntax into equivalent Python syntax. It handles Java-specific keywords, operators, and expressions, and converts them into Python equivalents.

Data Type Conversion: Java and Python have different data types and type systems. The converter takes care of converting Java data types, such as int, float, boolean, etc., into their Python equivalents, such as int, float, bool, etc.

**1.1 SUMMARY OF APPLICATION**

Java; Python is an easier language for programmers to learn. One can progress faster if learning programming in Python as a first language, because Java is restrictive and more complex compared to Python.

If programmers want to translate their software programs from Java to Python to gain its features, they will have to rewrite the whole program from start which consumes time and increases cost. Therefore, a mechanism that translates programs from Java to Python automatically is necessary.

Here, we develop a simple Java-Python translator that takes a Java file code as input and translates it to Python file code as output.In summary, a Java to Python converter is a useful tool for converting Java code into Python code. It simplifies the process of migrating projects or understanding the equivalent code in a different language. While converters can save time, it's important to review and refine the converted code manually to ensure it aligns with Python's idiomatic conventions and requirements.

**1.2 BACKGROUND OF APPLICATION**

Here, we develop a simple Java-Python translator that takes a Java file code as input and translates it to Python file code as output.

Extensive research and analysis were conducted to understand the existing market landscape and user requirements. User feedback, surveys, and usability studies played a crucial role in shaping the features and functionalities of the app.

The development process involved a multidisciplinary team of designers, developers, and usability experts who collaborated to create a seamless user experience. Attention was given to the app's interface, ensuring a clean and intuitive design that simplifies the process of setting and managing reminders.

Notifications and Alerts: The application can send notifications or alerts to relevant stakeholders, such as developers or project managers, when conversion processes are completed, errors are encountered, or specific milestones are reached. This helps to keep everyone informed about the progress and status of the Java to Python conversion.

**CHAPTER 2**

**REQUIREMENTS OF APPLICATION**

* **Software Requirements**
* HTML
* CSS
* Java Script
* Python
* **Hardware Requirements**
* A Laptop or desktop with following specifications -
* Processor:  i5
* RAM:  4GB
* Storage: 256GB

**CHAPTER 3**

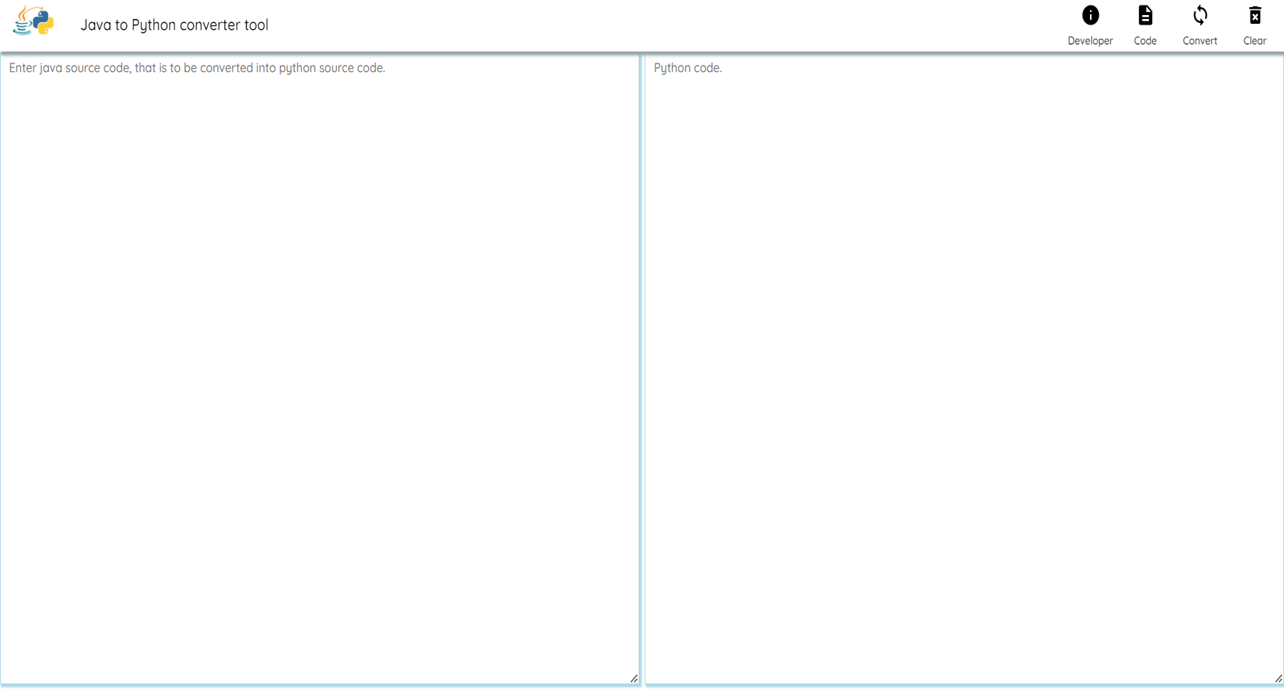
**FLOW CHART**

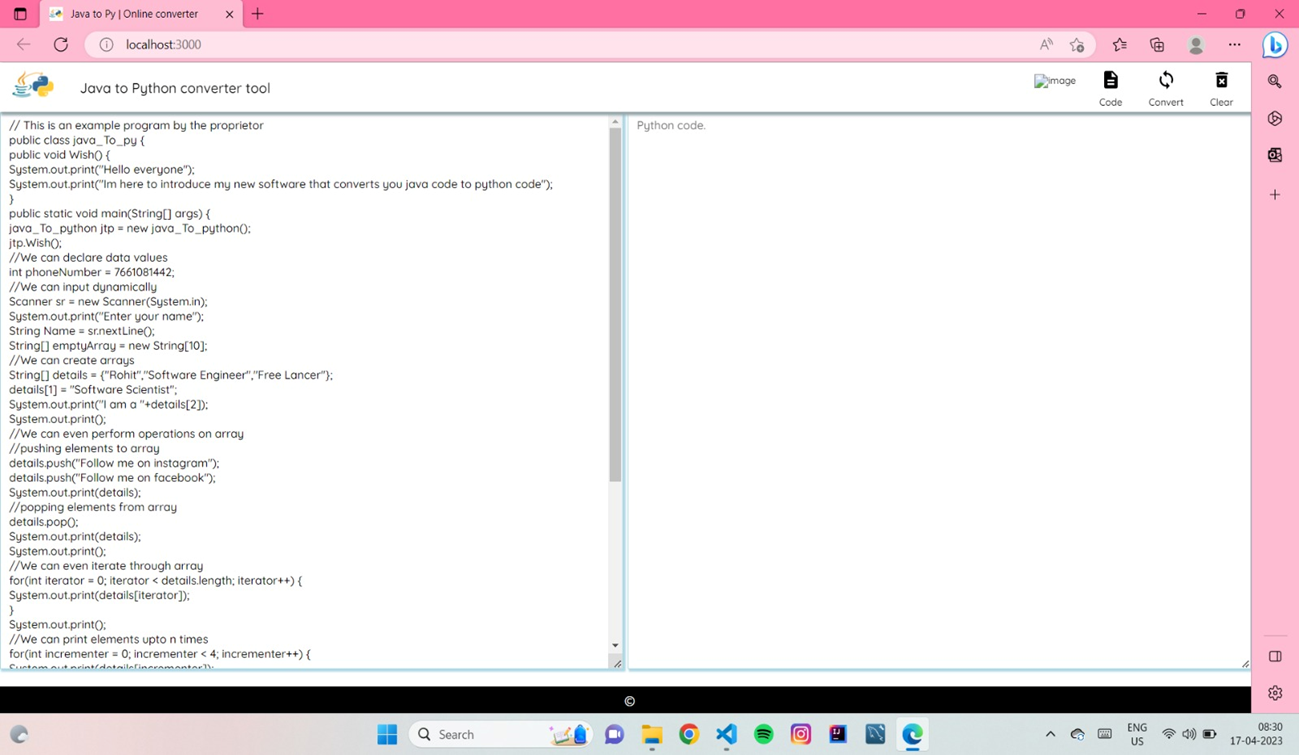
**Diagram

Description automatically generated**

Fig 3.1 Flow Chart

**3.1 SCREENSHOTS OF APP**

****



Graphical user interface, text, application

Description automatically generated

A picture containing text

Description automatically generated

(a). JAVA INPUT CODE (b).PYTHON OUTPUT CODE

**CHAPTER 4**

**APP’S CODE**

import React from 'react';

4

import './index.css';

5

/\*

6

\*\* Welcome to CIRL's new release, this is a software created by Rohit Acharya.

7

\*\* This software converts your java code into python.

8

\*\* All you need to do is write your java code and click on convert button, then your code

will be converted into python source code and will be pasted on right side of the page, in

python text area.

9

\*/

10

async function ConvertToPy() {

11

/\*

12

\*\* After clicking on the convert button, the code will be processed.

13

\*\* 1. Getting the raw text from the user and storing into the variable -> code.

14

\*\* 2. Parsing the code using L0 - Parser( ) methodology. i.e checking if all the

brackets and paranthesis are balanced or not.

15

\*/

16

var code = document.getElementById("code").value.trim();

17

var status = "";

18

// Parsing the code, if parsed successfully then only the code will be converted into

python

19

var Stack = [];

20

var stackPointer = 0;

21

// push method adds all the { [ ( into the stack

22

const push = (element) => {

23

Stack[stackPointer] = element;

24

stackPointer++;

25

}

26

// while popping all the { [ ( elements will be removed from Stack by receving ) ] }

Elements

27

const pop = () => {

28

try {

29

Stack[stackPointer - 1] = ``;

30

stackPointer--;

31

if (Stack[0] === ``) {

32

// Checking whether the Stack is empty or not ?

33

// if the stack is empty -> that means the code it perfectly balanced and

allowed to be converted to python code.

34

status = "true"

35

// Setting the status as true, to confirm that code is balanced precisely.

36

}

37

} catch (e) {

38

alert("Error : Parse UNSUCCESSFUL");

39

}

40

}

41

// Checking the code by converting the String into chunks of size 1

42

for (var ptr = 0; ptr <= code.length; ptr++) {

43

if (code[ptr] === '{' || code[ptr] === '(' || code[ptr] === '[') {

44

var sym = code[ptr];

45

push(sym);

46

} else if (code[ptr] === '}' || code[ptr] === ')' || code[ptr] === ']') {

[5:42 PM, 7/3/2023] NIKHIL Malla REDDY: pop();

48

}

49

}

50

// if the code is parsed then, its allowed to be converted into python

51

if (status === "true") {

52

var code = document.getElementById("code").value.trim();

53

var codeFragmentation = [];

54

var j = 0;

55

// First the code is coverted into chunks of space 1 and is being added to the

local storage.

56

// Two consecutives spaces arent allowed, if two consecutive spaces are

encountered then an exception is raised.

57

for (var data = 0; data < code.length; data++) {

58

if (code[data] !== null) {

59

if ((code[data] === " " && code[data + 1] === " ")) {

60

alert("Extra spaces arent allowed");

61

localStorage.clear();

62

break;

63

}

64

else {

65

// Adding code to localStorage by chunks of space one or by bits.

66

codeFragmentation[j] = code[data];

67

j++;

68

}

69

}

70

}

71

localStorage.setItem('code', JSON.stringify(codeFragmentation));

72

// Parsing

73

var print = "";

74

var id\_ = 0;

75

var functionName = "";

76

var function = [];

77

var className = "";

78

var name = [];

79

// Getting class name

80

for (var cN = 0; cN < localStorage.getItem("code").length; cN++) {

81

if ((JSON.parse(localStorage.getItem("code"))[cN] === "c" &&

JSON.parse(localStorage.getItem("code"))[cN + 1] === "l" &&

JSON.parse(localStorage.getItem("code"))[cN + 2] === "a" &&

JSON.parse(localStorage.getItem("code"))[cN + 3] === "s" &&

JSON.parse(localStorage.getItem("code"))[cN + 4] === "s" &&

JSON.parse(localStorage.getItem("code"))[cN + 5] === " ")) {

82

// The key word "class" is been recognized by the program.

83

for (var cName = cN + 5; cName < localStorage.getItem("code").length;

cName++) {

84

if (JSON.parse(localStorage.getItem("code"))[cName] !== "{") {

85

className +=

JSON.parse(localStorage.getItem("code"))[cName].trim();

86

}

87

if (JSON.parse(localStorage.getItem("code"))[cName] === "{") {

88

break;

89

}

90

}

91

}

92

}

93

name = className.split(" ");

94

className = name[0].trim();

95

for (var data = 0; data < localStorage.getItem("code").length; data++) {

96

// Processing syntax -> public void Method( );

97

if ((JSON.parse(localStorage.getItem("code"))[data] === `(` &&

JSON.parse(localStorage.getItem("code"))[data + 1] === `)` &&

JSON.parse(localStorage.getItem("code"))[data + 2] !== `;`) ||

(JSON.parse(localStorage.getItem("code"))[data] === `(` &&

JSON.parse(localStorage.getItem("code"))[data + 2] === `)` &&

JSON.parse(localStorage.getItem("code"))[data + 3] !== `;`)) { // No argument funtion

98

for (var ptr = data; ptr >= 0; ptr--) {

99

functionName += JSON.parse(localStorage.getItem("code"))[ptr];

100 }

101 function = functionName.split(" ");

102 functionName = function[0];

103 var function\_id = "";

104 for (var rev = functionName.length - 1; rev > 0; rev--) {

105 function\_id += functionName[rev];

106 }

107 localStorage.setItem(id\_, `def ${function\_id} () :`)

108 /\*

109 After setting the item to local storage, the local storage id is update by

one and all the variable values are flushed.

110 \*/

111 id\_++;

112 name = "";

113 function\_id = "";

114 functionName = "";

115 function = [];

116 }

117 //Processing print out method with out double quotes

118 else if ((JSON.parse(localStorage.getItem("code"))[data] === `p` &&

JSON.parse(localStorage.getItem("code"))[data - 1] === `.` &&

JSON.parse(localStorage.getItem("code"))[data + 1] === `r` &&

JSON.parse(localStorage.getItem("code"))[data + 2] === `i` &&

JSON.parse(localStorage.getItem("code"))[data + 3] === `n` &&

JSON.parse(localStorage.getItem("code"))[data + 4] === `t` &&

JSON.parse(localStorage.getItem("code"))[data + 5] !== `l`)) {

119 var print = "";

120 for (var ptr = [data + 5]; ptr < localStorage.getItem("code").length;

ptr++) {

121 if (JSON.parse(localStorage.getItem("code"))[ptr] !== ";") {

122 print += JSON.parse(localStorage.getItem("code"))[ptr];

123 }

124 if (JSON.parse(localStorage.getItem("code"))[ptr] === ";") {

125 localStorage.setItem(id\_, `\tprint${print}`);

126 id\_++;

127 print = "";

128 break;

129 }

130 }

131 }

132 // Processing syntax tocken -> public static void main(String[] args) = main

Method

133

else if (JSON.parse(localStorage.getItem("code"))[data] === `p` &&

JSON.parse(localStorage.getItem("code"))[data + 7] === `s` &&

JSON.parse(localStorage.getItem("code"))[data + 14] === 'v' &&

JSON.parse(localStorage.getItem("code"))[data + 19] === 'm') {

134

var mainMethod = "";

135

for (var ptr = data; ptr < localStorage.getItem("code").length; ptr++) {

136

if (JSON.parse(localStorage.getItem("code"))[ptr] !== `{`) {

137

mainMethod += JSON.parse(localStorage.getItem("code"))[ptr];

138

}

139

if (JSON.parse(localStorage.getItem("code"))[ptr] === `{`) {

140

if (mainMethod.includes("static") && mainMethod.includes("main"))

{

141

localStorage.setItem(id\_, `if \_name\_ == "\_main\_" :`);

142

}

143

mainMethod = "";

144

id\_++;

145

break;

146

}

147

}

148

var functionCalls = "";

149

for (var pt = data + 19; pt < localStorage.getItem("code").length; pt++) {

150

// Getting the object reference

151

if ((JSON.parse(localStorage.getItem("code"))[pt] === className[0]) &&

(JSON.parse(localStorage.getItem("code"))[pt + (className.length - 1)] ===

className[className.length - 1])) {

152

var syntax = "";

153

for (var Pt = pt; Pt < localStorage.getItem("code").length; Pt++)

{

154

// ClassName objRef = new ClassName();

155

// Consider the symbol "=" here and drive the code.

156

// Until the pointer reaches "=", concat all the chunks.

157

if (JSON.parse(localStorage.getItem("code"))[Pt] !== `=`) {

158

syntax += JSON.parse(localStorage.getItem("code"))[Pt];

159

}

160

if (JSON.parse(localStorage.getItem("code"))[Pt] === `=`) {

161

var object = "";

162

syntax = syntax.trim();

163

object = syntax.split(" ")[1].trim();

164

syntax = "";

165

break;

166

}

167

}

168

if (object !== "" && object !== null) {

169

for (var pointer = Pt; pointer <

localStorage.getItem("code").length; pointer++) {

170

if (JSON.parse(localStorage.getItem("code"))[pointer] ===

object[0] && JSON.parse(localStorage.getItem("code"))[pointer + (object.length - 1)] ===

object[object.length - 1] && JSON.parse(localStorage.getItem("code"))[pointer +

(object.length)] === `.`) {

171

for (var pointAt = [pointer + (object.length + 1)];

pointAt < localStorage.getItem("code").length; pointAt++) {

172

if

(JSON.parse(localStorage.getItem("code"))[pointAt] !== ';') {

173

functionCalls +=

JSON.parse(localStorage.getItem("code"))[pointAt];

[5:44 PM, 7/3/2023] NIKHIL Malla REDDY: }

175 if

(JSON.parse(localStorage.getItem("code"))[pointAt] === ';') {

176 localStorage.setItem(id\_,

`\t${(functionCalls)}`);

177 functionCalls = "";

178 object = "";

179 id\_++;

180 break;

181 }

182 }

183 }

184 }

185 }

186 }

187 }

188 }

189 // Processing Data types and values

190 else if (JSON.parse(localStorage.getItem("code"))[data] === `=` &&

JSON.parse(localStorage.getItem("code"))[data-2] !== `]` &&

JSON.parse(localStorage.getItem("code"))[data + 1] !== `=` &&

JSON.parse(localStorage.getItem("code"))[data + 2] !== `n` &&

JSON.parse(localStorage.getItem("code"))[data + 3] !== `e` &&

JSON.parse(localStorage.getItem("code"))[data + 4] !== `w`) {

191 //Hence its not a obj declairation, proved

192 //Getting object name

193 var objectName = "";

194 for (var objName = [data - 2]; objName >= 0; objName--) {

195 if (JSON.parse(localStorage.getItem("code"))[objName] !== " ") {

196 objectName +=

JSON.parse(localStorage.getItem("code"))[objName].replace("[", "").replace("]", "");

197 }

198 if (JSON.parse(localStorage.getItem("code"))[objName] === " " ||

JSON.parse(localStorage.getItem("code"))[objName] === "\n") {

199 var variableName = "";

200 for (var rev = objectName.length - 1; rev >= 0; rev--) {

201 variableName += objectName.charAt(rev);

202 }

203 break;

204 }

205 }

206 // Getting value

207 var objectValue = "";

208 for (var objData = [data + 2]; objData <=

localStorage.getItem("code").length; objData++) {

209 if (JSON.parse(localStorage.getItem("code"))[objData] !== `;`) {

210 objectValue +=

JSON.parse(localStorage.getItem("code"))[objData].replace("{", "[").replace("}", "]");

211 }

212 if (JSON.parse(localStorage.getItem("code"))[objData] === `;`) {

213 break;

214 }

215 }

216 if (objectName !== "" && objectValue !== "") {

[5:44 PM, 7/3/2023] NIKHIL Malla REDDY: if (objectValue.trim().includes(".nextLine") ||

objectValue.trim().includes(".nextInt")) {

218 // Dynamic input

219 objectValue = objectValue.replace("nextLine",

"input").replace("nextInt", "input");

220 localStorage.setItem(id\_, `\t${variableName.replace("(",

"").trim()} = ${objectValue.split(".")[1].trim()}`)

221 variableName = objectValue = "";

222 id\_++;

223 }

224 else {

225 // Static input

226 localStorage.setItem(id\_, `\t${variableName.replace("(",

"").trim()} = ${objectValue.trim()}`)

227 variableName = objectValue = "";

228 id\_++;

229 }

230 }

231 }

232 // Processing condition -> if( ) { }

233 else if ((JSON.parse(localStorage.getItem("code"))[data] === `i` &&

JSON.parse(localStorage.getItem("code"))[data + 1] === `f` &&

JSON.parse(localStorage.getItem("code"))[data - 1] === "\n") &&

(JSON.parse(localStorage.getItem("code"))[data + 2] === `(` ||

JSON.parse(localStorage.getItem("code"))[data + 3] === `(`)) { //if condition proved

234 var conditionStatement = "";

235 for (var ptr = data; ptr < localStorage.getItem("code").length; ptr++) {

236 if (JSON.parse(localStorage.getItem("code"))[ptr] !== `{`) {

237 conditionStatement +=

JSON.parse(localStorage.getItem("code"))[ptr];

238 }

239 if (JSON.parse(localStorage.getItem("code"))[ptr] === `{`) {

240 conditionStatement = conditionStatement.trim();

241 localStorage.setItem(id\_, `${conditionStatement} :`);

242 conditionStatement = "";

243 id\_++;

244 break;

245 }

246 }

247 }

248 // Processing condition -> else { }

249 else if (JSON.parse(localStorage.getItem("code"))[data] === `e` &&

JSON.parse(localStorage.getItem("code"))[data + 1] === `l` &&

JSON.parse(localStorage.getItem("code"))[data + 2] === `s` &&

JSON.parse(localStorage.getItem("code"))[data + 3] === `e` &&

JSON.parse(localStorage.getItem("code"))[data + 5] !== `i`) {

250 var conditionStatement = "";

251 for (var ptr = [data]; ptr < localStorage.getItem("code").length; ptr++) {

252 if (JSON.parse(localStorage.getItem("code"))[ptr] !== `{`) {

253 conditionStatement +=

JSON.parse(localStorage.getItem("code"))[ptr];

254 }

255 if (JSON.parse(localStorage.getItem("code"))[ptr] === `{`) {

256 conditionStatement = conditionStatement.trim();

257 localStorage.setItem(id\_, `${conditionStatement} :`);

conditionStatement = "";

259 id\_++;

260 break;

261 }

262 }

263 }

264 // Processing condition -> else if( ) { }

265 else if ((JSON.parse(localStorage.getItem("code"))[data] === `e` &&

JSON.parse(localStorage.getItem("code"))[data + 1] === `l` &&

JSON.parse(localStorage.getItem("code"))[data + 2] === `s` &&

JSON.parse(localStorage.getItem("code"))[data + 3] === `e` &&

JSON.parse(localStorage.getItem("code"))[data + 4] === ` ` &&

JSON.parse(localStorage.getItem("code"))[data + 5] === `i`)) {

266 var conditionStatement = "";

267 for (var ptr = [data + 7]; ptr < localStorage.getItem("code").length;

ptr++) {

268 if (JSON.parse(localStorage.getItem("code"))[ptr] !== `{`) {

269 conditionStatement +=

JSON.parse(localStorage.getItem("code"))[ptr];

270 }

271 if (JSON.parse(localStorage.getItem("code"))[ptr] === `{`) {

272 conditionStatement = conditionStatement.replace("else",

"").trim();

273 localStorage.setItem(id\_, `elif ${conditionStatement} :`);

274 conditionStatement = "";

275 id\_++;

276 break;

277 }

278 }

279 }

Index .css

{

margin: 0;

font-family: 'Quicksand', sans-serif;

margin-top: 1px;

}

body {

margin: 0;

font-family: 'Quicksand', sans-serif;

-webkit-font-smoothing: antialiased;

-moz-osx-font-smoothing: grayscale;

}

code {

font-family: 'Quicksand', sans-serif;

}

nav {

position: fixed;

width: 100%;

background-color: white;

box-shadow: 3px 2px 3px 2px rgb(149, 165, 170);

display: flex;

padding-left: 1%;

align-items: center;

top: 0;

margin-bottom: 0;

}

nav > p {

margin-left: 2%;

}

#logo {

width: 24px;

}

center {

margin-top: 1%;

}

tr > td{

font-size: .7rem;

align-items: center;

text-align: center;

width: 62px;

}

#Right-Nav {

right: 0;

position: fixed;

display: flex;

justify-content: center;

justify-items: center;

align-items: center;

margin-right: 1px;

}

#logo:hover{

background-color: rgb(149, 178, 211);

border-radius: 5px;

cursor: pointer;

}

#code {

padding: 5px;

width: 48.7%;

border: 1px solid lightblue;

box-shadow: 2px 2px 2px 1px lightblue;

outline: none;

padding-left: 10px;

height: 625px;

float: left;

}

#final\_code {

padding: 5px;

width: 48.7%;

border: 1px solid lightblue;

box-shadow: 2px 2px 2px 1px lightblue;

outline: none;

padding-left: 10px;

outline: none;

height: 625px;

float: right;

}

#btn {

width: 99%;

background-color: rgb(14, 87, 243);

outline: none;

border: 1px solid rgb(14, 87, 243);

box-shadow: 1px 1px 2px 2px rgb(24, 223, 223);

color: white;

}

#text {

font-size: 0.9rem;

}

#info{

position: fixed;

bottom: 0;

width: 100%;

background-color: black;

color: white;

padding: 5px;

}

#logoImg {

width: 50px;

}

#load {

position: fixed;

width: 40px;

margin-top: 10px;

margin-right: 18px;

right: 0;

visibility: hidden;

}

@media screen and (max-width:1308px) {

#code {

width: 48%;

}

#final\_code {

width: 48%;

}

#load {

margin-top: 15px;

}

}

@media screen and (max-width:700px) {

#code{

margin-top: 2%;

width: 97%;

height: 290px;

}

#final\_code{

width: 97%;

height: 250px;

}

#btn {

width: fit-content;

border-radius: 5px;

background-color: rgb(14, 87, 243);

outline: none;

border: 1px solid rgb(14, 87, 243);

box-shadow: 1px 1px 2px 2px rgb(24, 223, 223);

color: white;

}

#text {

font-size: 0.7rem;

}

}

@media screen and (max-width:500px) {

p {

font-size: 0.7rem;

}#title {

visibility: hidden;

}

#load {

position: fixed;

width: 20px;

margin-right: 18px;

right: 0;

bottom: 0;

margin-bottom: 50px;

visibility: hidden;

Index.js

import React from 'react';

import ReactDOM from 'react-dom';

import JavaToPy from './JavaToPy';

ReactDOM.render(

<React.StrictMode>

<JavaToPy />

</React.StrictMode>,

document.getElementById('root')

);

**4.1 Handling Errors**

Here are some general steps to handle errors effectively:

1. Identify Potential Error Points: Review your code and identify areas where errors could occur. These can include user input, file operations, database interactions, network requests, etc.
2. Implement Error Handling Mechanisms: Use try-except blocks to catch and handle specific types of exceptions that may occur. For example:

try:

# Code that may raise an exception

except ExceptionType:

# Handle the specific exception

Replace Exception Type with the specific type of exception you want to handle, such as Value Error, FileNotFoundError, Database Error, etc.

1. Display Meaningful Error Messages: When an exception is caught, display an informative error message to the user. Include details about the error and possible solutions. For example:

try:

# Code that may raise an exception

except ValueError:

print("Invalid input! Please enter a valid value.")

1. Logging: Implement logging mechanisms to record errors and exceptions. Use a logging library to write error logs to a file or a log management system. Include relevant information like the error message, timestamp, and any relevant context details.

import logging

logging.basicConfig(filename='error.log', level=logging.ERROR)

try:

# Code that may raise an exception

except Exception as e:

logging.error(str(e))

1. Graceful Degradation: Handle expected errors gracefully and provide fallback options or alternative paths. For example, if a network request fails, display a message to the user indicating the issue and providing alternative options if available.

6. Thorough Testing: Test your application thoroughly to identify and fix any errors or bugs before releasing it. Create test cases to cover different scenarios and edge cases. Use automated testing frameworks to perform unit tests, integration tests, and user acceptance tests.

**CHAPTER 5**

**CONCLUSION**

This paper has presented a simple Java Python translator that converts Java code to Python code using XML as an intermediate language.The translator covers class and method declaration, comments, declaring and initializing primitive, floating points and boolean variables, selection statements, switch statement and iteration Statements.

After designing, implementing and testing the translator, we found that it successfully converts the syntax of Java programs to Python without having to rewrite Python programs from start.However, it's important to note that while the Java to Python converter automates the process to a large extent, it may not be able to handle all cases perfectly. Java and Python are different languages with unique characteristics, and some aspects of the code may require manual intervention after the conversion. Developers should review and test the translated code to ensure its correctness and optimal performance.

Maintenance: It's important to consider the long-term maintenance and support of the converted Python code. As programming languages evolve, updates or modifications might be needed to keep the code up to date and compatible with the latest Python versions.

Testing and Debugging: The converted Python code should be thoroughly tested and debugged to ensure its correctness and functionality. Manual intervention might be necessary to fix any errors or inconsistencies introduced during the conversion process.

In conclusion, a Java to Python converter is a valuable tool that helps developers convert Java code into Python code. It simplifies the process of porting applications or projects from one programming language to another, enabling developers to leverage the strengths and features of Python

**CHAPTER 6**

**FUTURE SCOPE**

Improved Accuracy: Code converters can benefit from advancements in natural language processing (NLP) and machine learning techniques to enhance the accuracy of the conversion process.Advanced Language Features: Java and Python are constantly evolving, with new language features and constructs being introduced over time.

Support for Frameworks and Libraries: Java and Python ecosystems have rich sets of frameworks and libraries that developers rely on for various applications. This would facilitate the migration of complete applications, including their dependencies, from Java to Python.converters can also provide suggestions for optimizing the converted: In addition to converting the code, future Python code. This can involve identifying areas for performance improvement, recommending Pythonic coding patterns, or suggesting alternative Python libraries that align with the original Java code's functionality.

Support for Other Languages: While the focus of a Java to Python converter is to convert Java code, expanding its capabilities to support other programming languages can increase its versatility and usefulness. Adding support for converting code from languages like C, C++, or JavaScript would provide developers with a more comprehensive conversion toolset.

It's important to note that automatic code conversion between two languages can have limitations due to the fundamental differences between Java and Python. Some concepts or patterns may not have a direct equivalent, requiring manual adjustments and code rewriting in certain cases. Therefore, while a converter tool can assist in the initial migration, it's essential for developers to review and test the converted code thoroughly to ensure its correctness and maintainability.