1. INTRODUCTION

In the modern digital landscape, the need for efficient file storage, management, and transfer has become increasingly important. With the exponential growth in data generation and the widespread use of multimedia files, the challenges of handling large files have grown significantly. These large files, such as high-resolution images, videos, and lengthy documents, not only consume a substantial amount of storage space but can also be cumbersome to transfer across networks, especially in environments with limited bandwidth.

File compression and format conversion are essential techniques for addressing these issues. Compression reduces file sizes, making it easier to store, share, and transfer data while maintaining the integrity of the original content. Conversely, file format conversion ensures that files are compatible with different software and devices, providing greater flexibility for users.

The **File Compressor and Converter** tool was developed to meet these needs, offering a seamless solution to reduce the size of various types of files and to convert them between different formats. Whether it's compressing large video files for faster streaming or converting document formats for compatibility with different programs, this tool provides a versatile, user-friendly solution for individuals and businesses alike.

This tool supports a broad range of file formats, including images (e.g., JPG, PNG), videos (e.g., MP4, AVI), documents (e.g., DOCX, PDF), and audio (e.g., MP3, WAV), making it suitable for diverse user requirements. Through the use of advanced algorithms, such as Huffman coding for compression and FFmpeg for multimedia file conversions, the tool efficiently reduces file sizes without compromising quality and ensures that file conversion is carried out seamlessly, allowing users to work across different platforms.

The primary goals of the **File Compressor and Converter** tool are to help users save storage space, speed up file transfer processes, and facilitate the use of files in different formats across various applications and devices. By providing an easy-to-use interface and robust functionality, this tool aims to streamline digital workflows and improve overall productivity for both personal and professional use.

1.1. Type of File Compressor and Converter System

1.2. Problem Statement

In today's digital era, the rapid growth of data and the need to store and share files efficiently have become increasingly important. Files, particularly images, documents, and videos, often take up significant storage space, which can lead to problems in terms of device storage limitations and longer transmission times, especially over networks with limited bandwidth.

Additionally, users often need to work with different file formats, requiring conversions for compatibility across various platforms, software, or devices. The lack of easy-to-use, efficient tools that combine both compression and conversion functionalities for multiple file types adds complexity to the process.

Currently, most systems offer either compression or conversion separately, leading to inefficiencies for users who need both. Existing tools may also lack user-friendly interfaces, support for multiple file formats, or sufficient compression ratios, ultimately making the process more time-consuming and cumbersome.

Therefore, there is a need for an integrated **File Compressor and Converter System** that can:

- Compress various file types (such as documents, images, videos) to reduce their size while maintaining acceptable quality.
- Convert files from one format to another, ensuring compatibility across different platforms and devices.
- Provide a user-friendly interface, allowing users to compress and convert files quickly and efficiently with minimal technical knowledge.

This project aims to design and develop such a system that addresses these issues, providing users with a reliable, fast, and easy-to-use solution to handle file compression and conversion in one integrated platform.

1.3.Objectives

The "File Compressor and Converter" platform aims to achieve the following objectives:

1. Develop a File Compression Module:

- Implement efficient algorithms to compress various types of files (e.g., images, documents, videos) while maintaining acceptable quality.
- Ensure the compression process results in significant space-saving without compromising the integrity of the file.

2. Create a File Conversion Module:

- Design the system to support the conversion of multiple file formats (e.g., .jpg to .png, .docx to .pdf, .mp4 to .avi).

- Ensure that the conversion process maintains the original content and formatting of the file, Implement features such as batch processing to handle multiple files at once, saving time for the user.

3. Ensure Cross-Platform Compatibility:

- Design the system to work across multiple operating systems (Windows, Linux, macOS), ensuring it is accessible to a wide range of users.
- Support a variety of file formats commonly used in different industries or everyday applications.

4. Support for Multiple File Types:

- The system should support a wide range of file types, including but not limited to text allowing seamless interoperability across platforms.

5. Implement a User-Friendly Interface:

- Develop an easy-to-use graphical user interface (GUI) or command-line interface (CLI) that simplifies the process for users to select, compress, and convert files.
- Ensure the interface is intuitive, with clear instructions and minimal user input required for performing tasks.

6. Optimize Performance and Efficiency:

- Ensure that both the compression and conversion processes are optimized for speed and resource utilization, even when handling large files, documents, audio, video, images, and compressed archives (e.g., .zip, .rar).

7. Achieve High Compression Ratios:

- Utilize efficient compression techniques to ensure that the system can achieve significant reductions in file size without causing notable degradation in quality or usability of the file.

8. Provide Secure File Handling:

- Implement safety mechanisms to ensure that files are handled securely during compression and conversion processes.
- Ensure that no data loss occurs and that the file remains intact and usable after operations.

9. Offer Detailed Feedback and Logging:

- Provide real-time feedback to the user during compression and conversion processes (e.g. progress bars, time estimates).
- Log system errors and success messages for users, allowing them to track the status of their file operations.

10. Test and Evaluate the System's Effectiveness:

- Conduct extensive testing on different file types to evaluate the performance and reliability of both compression and conversion processes.
- Collect and analyze user feedback to continuously improve the system.

1.4. Background of Study

In the digital age, the storage and transmission of large amounts of data have become essential for businesses, individuals, and organizations. As file sizes continue to grow due to high-resolution images, videos, and data-intensive applications, there is an increasing need for efficient ways to manage, store, and share files. Two crucial techniques in addressing these needs are file compression and file conversion. File Compression refers to the process of reducing the size of files or data without losing important information. Compression algorithms use various techniques such as encoding, lossless methods, or lossy techniques to shrink files. Compressed files consume less storage space and can be transmitted over networks faster, making them ideal for a wide range of applications, from cloud storage to email attachments and web services.

On the other hand, File Conversion involves changing a file's format from one type to another to ensure compatibility across different platforms and systems. For instance, converting video files from one format (e.g., MP4) to another (e.g., AVI) or converting a text file from one encoding format to another. File conversion is important in a world with numerous file formats, allowing users to work across different software applications that support various formats.

Combining compression and conversion into a single tool offers significant advantages. A file compressor and converter can not only reduce file size but also enable users to convert files into different formats for better compatibility and easier sharing. This can be particularly valuable for professionals in areas like media production, software development, cloud computing, and more.

While file compressors and converters are widely available, most existing tools focus on only one of these processes. This study aims to explore the development of an integrated solution that combines both functionalities. By understanding how compression and conversion techniques can be optimized, we can create a tool that meets modern data management needs in an efficient and user-friendly manner.

2. METHODOLOGY

1.4.File Compressor and Convertor System Overview

A **File Compressor and Converter System** is a software tool that allows users to compress files into smaller sizes to save storage space and convert files from one format to another. This system can be applied to a wide range of file types, including images, audio, video, text, and documents. It is commonly used in data management, storage optimization, and improving transfer speeds.

1.6. Working

1.7. System Components

The compression process reduces the file size by removing redundancies and utilizing algorithms to condense the data. Common file compression formats include:

- **ZIP:** General-purpose compression that supports multiple file types.
- **RAR:** Often used for large files and directories, with strong compression algorithms.
- **GZIP:** Used mainly for text files like HTML, JavaScript, and CSS.
- **7z:** A high-compression format, often used for large files.

Compression techniques:

- **Lossless Compression:** Reduces file size without losing any data (e.g., ZIP, PNG).
- **Lossy Compression:** Achieves higher compression rates by removing some of the file's data, typically used for audio and video files (e.g., MP3, MP4, JPEG).

1.8. Functionality

The **File Compressor and Converter System** can have various functionalities to serve the purpose of compressing and converting files efficiently. Here's a breakdown of the core functionalities that your system should support:

1.8.1. File Upload and Input Handling

- **File Upload**: Users can select files from their local system or drag and drop files into the application interface.
- **Multiple File Types**: The system should support multiple types of files (e.g., images, text documents, videos, audio files, etc.).
- **Batch Upload**: Users can select and upload multiple files at once for batch compression or conversion.

1.8.2. File Compression

- **Select Compression Format**: Users can choose from different compression formats such as ZIP, GZIP, or TAR.
- **Compression Level Control**: Allow users to control the compression level (e.g., high, medium, or low compression).
- **Real-Time Compression Progress**: Display a progress bar or status indicating the file compression process.
- **Download Compressed File**: After compression, the system should allow users to download the compressed file.

1.8.3. File Conversion

- **Select Conversion Format**: Users can select the desired output format for their files. For example:
    - **Images**: JPG to PNG, BMP to JPEG, etc.

- **Audio**: WAV to MP3, FLAC to MP3, etc.
- **Video**: AVI to MP4, MKV to MP4, etc.
- **Text Documents**: DOCX to PDF, TXT to HTML, etc.

1.9. Algorithm/ Flowchart

1.9.1. Algorithm:

Step 1: Register the user with an email, and verify the account using OTP.
Step 2: Log in with credentials (email/username/mobile).
Step 3: Complete the profile by entering details like name, age, travel preferences, etc.
Step 4: Use the buddy matching system to either manually select a destination or use the map for location-based matching.
Step 5: Find travel buddies within a 5KM radius or based on the chosen destination.
Step 6: Add buddies to a travel group.
Step 7: Activate the trip and manage group details.

- Step 8:

8: Save all travel details and wait for a notification when buddies are found.

1.9.2.Flowchart:
6StartLogged in?Main PageHome PageLoginRegistrationSome PlacesMapProfile PageCreateGroupUpdate ProfileFind BuddyNotification of Buddy FoundOTP VerificationAdd MemberLocationEnter ManualySelect on MapEndEndYesNoYes
No

1.9.3. Program:

```
from flask import Flask, request, jsonify, send_from_directory
from pymongo import MongoClient
import os
import logging
from werkzeug.security import generate_password_hash, check_password_hash
from werkzeug.utils import secure_filename
from dotenv import load_dotenv
from flask_cors import CORS
load_dotenv() # Load environment variables from .env file
# Logging configuration
logging.basicConfig(level=logging.DEBUG)
# Flask app configuration
app = Flask(__name__, static_folder='../public', template_folder='../public')
CORS(app)
# MongoDB connection
client = MongoClient('mongodb://localhost:27017/')
db = client['buddy_make_a_plan'] # Replace with your actual database name
users_collection = db['users']
group_collection = db['groups']
profile_collection = db[ ‘profiles'] # Define profile_collection here
# Serve static assets
@app.route('/css/<path:filename>')
def serve_css(filename):
```

```
return send_from_directory(os.path.join(app.static_folder, 'css'), filename)
@app.route('/js/<path:filename>')
def serve_js(filename):
return send_from_directory(os.path.join(app.static_folder, 'js'), filename
@app.route('/images/<path:filename>')
def images(filename):
return send_from_directory(os.path.join(app.static_folder, 'images'), filename)
@app.route('/<path:path>')
def send_static(path):
return send_from_directory('public', path)
@app.route('/')
def main():
return send_from_directory(app.static_folder, 'index.html'
# User registration
@app.route('/register', methods=['POST'])
def register():
data = request.json
username = data.get('username')
email = data.get('email')
password = data.get('password')
if users_collection.find_one({"username": username}) or users_collection.find_one({"email": email}):
return jsonify({"message": "Username or email already exists."}), 400
hashed_password = generate_password_hash(password)
user = {
```

```
"username": username,
"email": email,
"password": hashed_password
}
users_collection.insert_one(user)
return jsonify({"message": "Registration successful!"}), 201
# User login
@app.route('/login', methods=['POST'])
def login():
data = request.json
username = data.get('username')
password = data.get('password')
user = users_collection.find_one({"username": username})
if user and check_password_hash(user['password'], password):
return jsonify({"message": "Login successful!", "token": "fake-token"}), 200
else:
return jsonify({"message": "Invalid username or password."}), 401
# Get profile data
@app.route('/api/get-profile-data', methods=['GET'])
def get_profile_data():
# Replace this with actual user ID from session or token
user_id = request.args.get('user_id') # Get user_id from query params or session
user_data = profile_collection.find_one({'user_id': user_id}, {'_id': 0})
if user_data:
```

```
return jsonify(user_data)
else:
return jsonify({'message': 'Profile data not found'}), 40
# Add member to a group
@app.route('/api/add-member', methods=['POST'])
def add_member():
data = request.json
group_name = data.get('groupName')
if not group_name:
return jsonify({'message': 'Group name is required'}), 400
# Add member data to the group
group_collection.update_one(
{'group_name': group_name},
{'$push': {'members': data}},
upsert=True
)
return jsonify({'message': 'Member added successfully'}), 201
# Find buddies based on location
@app.route('/api/find-buddies', methods=['GET'])
def find_buddies():
latitude = float(request.args.get('latitude'))
longitude = float(request.args.get('longitude'))
# Find nearby buddies within a 10KM radius (modify this query for actual location filtering)
nearby_buddies = profile_collection.find({
```

```
'location': {
'$geoWithin': {
'$centerSphere': [[longitude, latitude], 10 / 6378.1]
}
}
}, {'_id': 0, 'first_name': 1, 'last_name': 1})
buddies = list(nearby_buddies)
return jsonify({'buddies': buddies, 'latitude': latitude, 'longitude': longitude, 'message': 'Coordinates received'})
# Serve HTML pages
@app.route('/add-buddy.html')
def add_buddy():
return send_from_directory(app.static_folder, 'add-buddy.html')
@app.route('/other.html')
def other():
return send_from_directory(app.static_folder, 'other.html')
@app.route('/profile.html')
def profile():
return send_from_directory(app.static_folder, 'profile.html')
@app.route('/review.html')
def review():
return send_from_directory(app.static_folder, 'review.html')
@app.route('/t&c.html')
def conditions():
return send_from_directory(app.static_folder, 't&c.html')
```

```
@app.route('/update-profile.html')
def update_profile_page():
return send_from_directory(app.static_folder, 'update-profile.html')
@app.route('/home.html')
def home():
return send_from_directory(app.static_folder, 'home.html')
@app.route('/head.html')
def header():
return send_from_directory(app.static_folder, 'head.html')
# Configure where to save the uploaded images
UPLOAD_FOLDER = 'uploads/profile_pics/'
os.makedirs(UPLOAD_FOLDER, exist_ok=True) # Ensure the upload directory exists
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
ALLOWED_EXTENSIONS = {'png', 'jpg', 'jpeg', 'gif'}
# Helper function to check file extension
def allowed_file(filename):
return '.' in filename and filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS
@app.route('/profile', methods=['GET'])
def get_profile():
token = request.headers.get('Authorization')
# Verify token and fetch user data from the database
if token: # Add proper token verification here
user = {
"username": "john_doe",
```

```
"name": "John Doe",
"email": "john@example.com",
"gender": "Male",
"mobile": "1234567890",
"about": "A brief description",
"photo": None # You can put the URL to the image here if needed
}
return jsonify(user), 200
else:
return jsonify({"error": "Unauthorized"}), 401
# Update profile route (uploading profile picture)
@app.route('/profile/update', methods=['POST'])
def update_profile():
# Get user_id from form data
user_id = request.form.get('userId') # Matches 'userId' from the frontend
if not user_id:
return jsonify({'error': 'User ID is required'}), 400
# Check if the file part is present
if 'photo' not in request.files: # Ensure 'photo' matches the frontend key
return jsonify({'error': 'No file part in request'}), 400
file = request.files['photo'] # Access the uploaded file
# Check if a file was selected
if file.filename == '':
return jsonify({'error': 'No file selected'}), 400
```

```
# Validate file type and save if allowed
if file and allowed_file(file.filename):
filename = secure_filename(file.filename)
filepath = os.path.join(app.config['UPLOAD_FOLDER'], filename)
file.save(filepath)
# Update other profile fields sent in the form
name = request.form.get('name')
email = request.form.get('email')
mobile = request.form.get('mobile')
gender = request.form.get('gender')
about = request.form.get('about')
# Update the user's profile in the database
profile_update = {
'profile_image': filepath, # Store the image path
'name': name,
'email': email,
'mobile': mobile,
'gender': gender,
'about': about
}
# Update the user's profile in the database (assuming MongoDB, adjust query accordingly)
result = profile_collection.update_one({"user_id": user_id}, {"$set": profile_update})
if result.modified_count == 1:
return jsonify({'message': 'Profile updated successfully!'}), 200
```

```
else:
return jsonify({'error': 'Profile update failed!'}), 400
return jsonify({'error': 'File type not allowed'}), 400
if __name__ == '__main__':
app.run(debug=True)
```

1.10. RESULT ANALYSIS

We developed the "Buddy Make a Plan" application as a web application. This platform allows users to register, create detailed profiles, and find potential travel companions based on common preferences like destination, age, and travel interests. It integrates a user-friendly interface for profile management and group creation, helping users find like-minded travel partners conveniently.

3. TOOLS AND TECHNOLOGIES

1.11. Hardware Configuration

- Processor: Intel® Core™ i5-8250U CPU @ 1.6GHz or above
- RAM: 4 GB RAM or above
- Storage: 500 MB or above
- Active Internet Connection

1.12. Software Requirements

- Python (Backend Logic)
- MongoDB (Database)
- Flask (Server Framework)
- Windows OS

1.13. Development Tools

1.13.1. Visual Studio Code (VS Code):

Visual Studio Code (VS Code) is a lightweight but powerful source code editor that supports a wide range of programming languages, including Python. Developed by Microsoft, VS Code provides an efficient and customizable environment for developing the "Buddy Make a Plan" project.

Key Features of VS Code:

- Code Editor: VS Code offers a highly customizable code editor with features such as syntax highlighting, intelligent code completion, and error detection. It also includes extensions that add support for different languages and tools.
- Integrated Terminal: The integrated terminal in VS Code allows developers to run scripts and manage environments directly from the editor without switching to an external terminal.
- Debugger: VS Code includes built-in debugging tools that support Python. It allows developers to set breakpoints, step through code, inspect variables, and evaluate expressions easily.
- Extensions Marketplace: VS Code has an extensive extensions marketplace that allows developers to add new capabilities, such as linting, version control, Docker integration, and more. Python extensions, like the official Python extension by Microsoft, help improve development efficiency.
- Version Control Integration: VS Code seamlessly integrates with version control systems like Git, allowing developers to manage branches, stage changes, and commit updates directly from the editor.
- Live Share: The Live Share feature allows developers to collaborate in real-time, making it easy for team members to share their development environment for pair programming or review sessions.

- Customization: VS Code provides themes, keybindings, and a variety of settings to customize the editor's look and behavior. It allows for a tailored development experience according to individual preferences.
- Cross-Platform Support: VS Code runs on Windows, macOS, and Linux, offering a consistent experience across various operating systems.

Additional Tools:
- MongoDB Atlas: Cloud-based MongoDB service used for managing the application database. Postman: API testing tool for debugging and testing API calls. Git: Version control system used for managing source code and collaboration.

1.14. Programming Languages

1.14.1. Python

Python is a high-level, interpreted programming language known for its simplicity, readability, and ease of use. Its clean syntax and dynamic typing make it a popular choice for both beginners and experienced developers working on projects of any scale. Python's object-oriented approach and language constructs facilitate the writing of logical and maintainable code, making it a versatile choice for the "File Compressor and Convertor system" project.

Key Features of Python:

Readable and Concise Syntax: Python emphasizes code readability which allows developers to write clean and easily understandable code. This emphasis on readability is supported by significant indentation that forms part of the language's syntax.
- Dynamic Typing and Memory Management: Python uses dynamic typing, which allows variables to change types during runtime. It also features automatic memory management through a combination of reference counting and a cycle-detecting garbage collector.
- Multiple Programming Paradigms: Python supports various programming paradigms, including procedural, object-oriented, and functional programming.
- Extensive Standard Library: Python is known as a "batteries included" language because of its comprehensive standard library.
- Zen of Python (PEP 20): Python's core philosophy is outlined in "The Zen of Python," which promotes simplicity and readability.
- Functional Programming Support: Python offers functional programming features such as map, filter, and reduce functions, list comprehensions, and generator expressions.
- Community and Extensibility: Python has a large community, with a rich ecosystem of libraries and frameworks. Modules like tkinter for GUI development can be installed easily using pip.

1.14.2. JavaScript (JS)

JavaScript is a lightweight, interpreted programming language used for building dynamic content on web pages. JavaScript is essential for creating interactive and responsive user interfaces for the "Buddy Make a Plan" platform, making it an integral part of the frontend development.

Key Features of JavaScript:

- Client-Side Scripting: JavaScript runs directly in the user's browser, making it ideal for client-side validation and creating dynamic page elements.
- Event-Driven Programming: JavaScript supports event-driven programming, enabling user interactions like clicks, form submissions, and keypresses to trigger specific functions.
- DOM Manipulation: JavaScript allows developers to manipulate the Document Object Model (DOM), making it easy to modify HTML elements, styles, and content dynamically.
- Asynchronous Programming: JavaScript supports asynchronous programming with callbacks, promises, and async/await, enabling better performance for web applications by allowing non-blocking operations.
- Versatile Language: JavaScript can be used on the client side for user interfaces and on the server side with platforms like Node.js, making it highly versatile.

4. Planning:

The total time taken for the complete implementation of our project was about 120 days which roughly measure up to almost 4 months. The complete project underwent the phases of requirement gathering and data analysis, code testing application testing, stabilization, performance enhancing, documentation and finally deployment.

July	August							September				October			
Analysis															
Design															
Coding															
Testing															
Implement															
W1	W2	W3	W4	W1	W2	W3	W4	W1	W2	W3	W4	W1	W2	W3	W4