

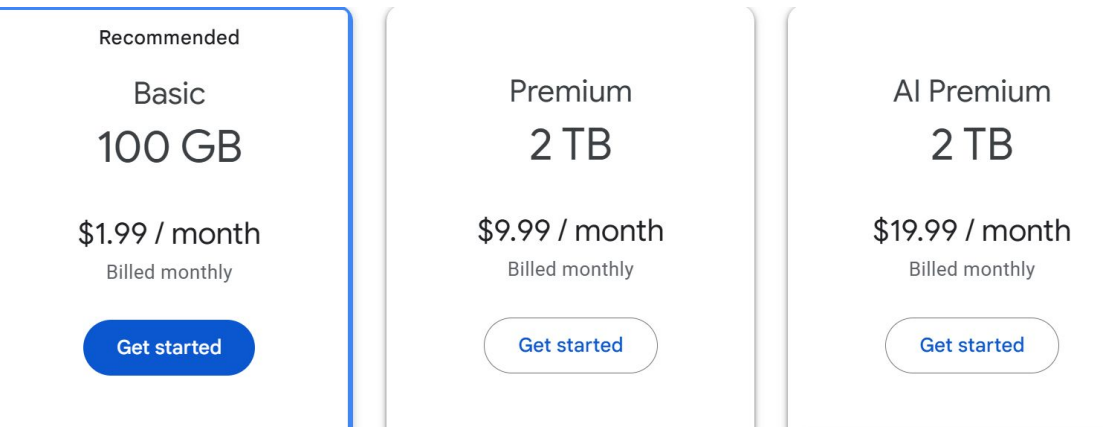
EcoPix



Junyu Li

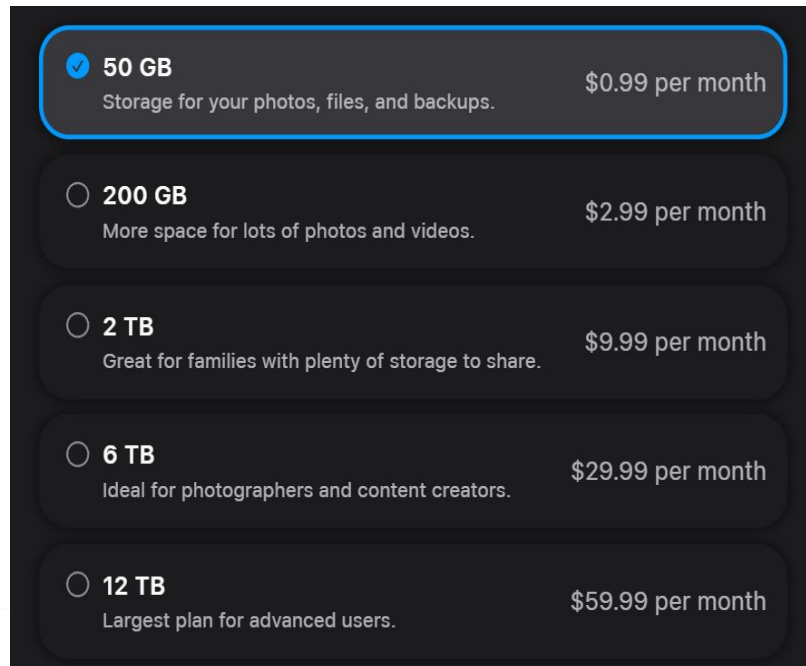
Issues

- High cost on subscription fee for a large amount of storage.
- Idle computers are common



A comparison of Google One storage plans. Three plans are shown side-by-side: Basic 100 GB, Premium 2 TB, and AI Premium 2 TB. The Basic plan is highlighted with a blue border and a 'Recommended' label. Each plan includes a 'Get started' button.

Plan	Storage	Price	Billing
Basic	100 GB	\$1.99 / month	Billed monthly
Premium	2 TB	\$9.99 / month	Billed monthly
AI Premium	2 TB	\$19.99 / month	Billed monthly



A comparison of Google One storage plans. Five plans are shown in a dark-themed interface. The 50 GB plan is selected with a blue checkmark and a blue border. Each plan includes a description and a 'Get started' button.

Plan	Storage	Description	Price
<input checked="" type="radio"/>	50 GB	Storage for your photos, files, and backups.	\$0.99 per month
<input type="radio"/>	200 GB	More space for lots of photos and videos.	\$2.99 per month
<input type="radio"/>	2 TB	Great for families with plenty of storage to share.	\$9.99 per month
<input type="radio"/>	6 TB	Ideal for photographers and content creators.	\$29.99 per month
<input type="radio"/>	12 TB	Largest plan for advanced users.	\$59.99 per month

Idea

- Personal Cloud-Based Photo Management and Viewing Platform
 - Personal Host Server
 - Light Weight on the Server-end
 - Easy to deployment

UI - Keywidget

- GridView - Photo gallery
- photo_detail - build the photo viewing
- db_helper - Embedded database for store sessions
- FlutterMap - Show maps
- Flutter_map_marker_cluster - Make easy to display makers on Fluttermap
- buildImageFromCookie - Minimise the connection
- ScrollController - Detect photo on screen location

Server

```
72 @routes.route('/photo/list', methods=['GET'])
73 def photo_list():
74     if 'username' in sess
75         stuff = get_photo
76         print(stuff)
77         return stuff
78     else:
79         #print("The sessi
80         return jsonify({"
81
82 #dont forget to chagne ge
83 @routes.route('/pic/thumb
84 def serve_thumbnail(filen
85     """
86     Serve a thumbnail ima
87     """
88     if 'username' in sess
89         try:
90             return send_f
91         except FileNotFou
92             return jsonif
93     else:
94         #print("The sessi
95         return jsonify({"
96
97
```

docker.desktop

PERSONAL

Search

🔍

?

🔔

🎓

⚙️

🗖️

J

More Docker. Easy Access. New Streamlined Plans. [Learn more](#)

✕

Containers

Images

Volumes

Builds

Docker Scout

Extensions

Containers

[Give feedback](#)

Container CPU usage ⓘ
No containers are running.

Container memory usage ⓘ
No containers are running.

Show charts

Search

☰

Only show running containers

<input type="checkbox"/>	Name	Container ID ↑	Image	Port(s)	CPU (%)	Last sta	Actions
<input type="checkbox"/>	photo-db	4842c1325aa7	mariadb:lat	13316:3306	N/A	34 minut	<div>▶ ⋮ 🗑️</div>
<input type="checkbox"/>	ecopix_docker	af7356ad33da	junyu07/ec	15381:15381	N/A	2 hours	<div>▶ ⋮ 🗑️</div>

Showing 2 items

Walkthroughs

✕

Resource Saver mode

▶ ⋮

RAM 0.92 GB CPU 0.00% Disk 56.63 GB avail. of 62.67 GB

Terminal v4.36.0

Server

```
72 @routes.route('/photo/list', methods=['GET'])
73 def photo_list():
74     if 'username' in session:
75         stuff = get_photo_list()
76         print(stuff)
77         return stuff
78     else:
79         #print("The session id is")
80         return jsonify({"message": "Unauthorized"}), 401
81
82 #dont forget to chagne get list to the actual website
83 @routes.route('/pic/thumbnaill/<path:filename>', methods=['GET'])
84 def serve_thumbnail(filename):
85     """
86     Serve a thumbnail image from the thumbnail directory.
87     """
88     if 'username' in session:
89         try:
90             return send_from_directory(THUMBNAIL_DIR, filename, as_attachment=False)
91         except FileNotFoundError:
92             return jsonify({"message": "File not found"}), 404
93     else:
94         #print("The session id is")
95         return jsonify({"message": "Unauthorized"}), 401
96
97
```

The screenshot shows the Docker Desktop application window. At the top, there's a navigation bar with icons for containers, images, volumes, and settings. Below this, a section titled 'Containers' displays a table of running containers. The table has columns for 'Name', 'CPU (%)', 'Last state', and 'Actions'. Two containers are listed: '16:3306' and '81:15381'. The '16:3306' container is in a 'Running' state, while '81:15381' is in a 'Exited' state. Below the table, there's a 'Showing 2 items' indicator and a close button. The bottom status bar shows 'Resource Saver mode' and system metrics: 'RAM 0.92 GB', 'CPU 0.00%', and 'Disk 56.63 GB avail. of 62.67 GB'.

Name	CPU (%)	Last state	Actions
16:3306	N/A	34 minut	▶ ⋮ 🗑
81:15381	N/A	2 hours a	▶ ⋮ 🗑

Database

Containers / photo-db

photo-db



4842c1325aa7



[mariadb:latest](#)

13316:3306

STATUS

Exited (0) (18 minutes ago)



Logs

Inspect

Bind mounts

Exec

Files

Stats

```
2024-12-04 00:19:35 2024-12-04 5:19:35 0 [Note] Plugin 'wsrep-provider' is disabled.
2024-12-04 00:19:35 2024-12-04 5:19:35 0 [Note] InnoDB: Buffer pool(s) load completed at 241204 5:19:35
2024-12-04 00:19:36 2024-12-04 5:19:36 0 [Note] Server socket created on IP: '0.0.0.0'.
2024-12-04 00:19:36 2024-12-04 5:19:36 0 [Note] Server socket created on IP: '::'.
2024-12-04 00:19:36 2024-12-04 5:19:36 0 [Note] mariabdb: Event Scheduler: Loaded 0 events
2024-12-04 00:19:36 2024-12-04 5:19:36 0 [Note] mariabdb: ready for connections.
2024-12-04 00:19:36 Version: '11.6.2-MariaDB-ubu2404' socket: '/run/mysqld/mysqld.sock' port: 3306 mariadb.org binary d
istribution
2024-12-04 00:27:29 2024-12-04 5:27:29 5 [Warning] Access
2024-12-04 00:37:25 2024-12-04 5:37:25 0 [Note] mariabdb
2024-12-04 00:37:25 2024-12-04 5:37:25 0 [Note] InnoDB: F
2024-12-04 00:37:25 2024-12-04 5:37:25 0 [Note] InnoDB: S
2024-12-04 00:37:25 2024-12-04 5:37:25 0 [Note] InnoDB: D
2024-12-04 00:37:25 2024-12-04 5:37:25 0 [Note] InnoDB: B
2024-12-04 00:37:25 2024-12-04 5:37:25 0 [Note] InnoDB: R
2024-12-04 00:37:25 2024-12-04 5:37:25 0 [Note] InnoDB: S
87
2024-12-04 00:37:25 2024-12-04 5:37:25 0 [Note] mariabdb:
2024-12-04 00:37:25
```

EcoPix_Docker > Docker > **\$ dbrun.sh**

Junyu Li, 4 days ago | 1 author (Junyu Li)

```
1 docker run --name photo-db \ Junyu Li, 4 days ago • db
2 -e MYSQL_ROOT_PASSWORD=rootpassword \
3 -e MYSQL_DATABASE=photo_app \
4 -e MYSQL_USER=user \
5 -e MYSQL_PASSWORD=password \
6 -p 13316:3306 \
7 -v /Users/teriri/WIP_CODE/CSC184/PhotoApp/Database:/var/lib/mysql \
8 -d mariadb:latest
```

Database

```
class Photo(db.Model):
    __tablename__ = 'photos' # Corrected to __tablename__

    id = db.Column(db.Integer, primary_key=True)
    filename = db.Column(db.String(255), nullable=False) # Photo filename
    filepath = db.Column(db.String(1024), unique=True, nullable=False) # Full file path
    folder_path = db.Column(db.String(1024)) # Folder organization (optional)
    thumbnail_path = db.Column(db.String(1024)) # Path to the generated thumbnail
    creation_date = db.Column(db.DateTime, default=datetime.utcnow) # Automatically set to current time
    gps_latitude = db.Column(db.Float, nullable=True) # GPS latitude metadata
    gps_longitude = db.Column(db.Float, nullable=True) # GPS longitude metadata
    camera_model = db.Column(db.String(255)) # Camera model metadata
    focal_length = db.Column(db.Float) # Focal length metadata
    lens_model = db.Column(db.String(255)) # Lens model metadata

    # Relationship to the Album model
    album_id = db.Column(db.Integer, db.ForeignKey('album.id'), nullable=True)
    album = db.relationship('Album', backref='photos', lazy=True)

    gps_cluster_id = db.Column(db.Integer, db.ForeignKey('gps_clusters.id'), nullable=True)
```

```
class GPSCluster(db.Model):
    __tablename__ = 'gps_clusters'

    id = db.Column(db.Integer, primary_key=True)
    cluster_latitude = db.Column(db.Float, nullable=False)
    cluster_longitude = db.Column(db.Float, nullable=False)
    photo_count = db.Column(db.Integer, nullable=False)
```

```
class Album(db.Model):
    __tablename__ = 'album' # Corrected to __tablename__

    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(255), nullable=False, unique=True) # Album name must be unique
    description = db.Column(db.Text, nullable=True) # Optional album description
    creation_date = db.Column(db.DateTime, default=datetime.utcnow) # Automatically set to current time
```


Demo

Bibliography

- Flutter Official Library <https://docs.flutter.dev/>
- SQLite Official Library <https://pub.dev/packages/sqlite>
- Cupertino_http Library https://pub.dev/packages/cupertino_http
- Flutter_map Library https://pub.dev/packages/flutter_map
- Latlong2 Library <https://pub.dev/packages/latlong2>
- Flutter_map_marker_cluster Library
https://pub.dev/packages/flutter_map_marker_cluster
- ChatGPT - For debug & implementation <https://chatgpt.com/>
- Flask Documentation <https://flask.palletsprojects.com/en/stable/>
- Docker Hub Documentation <https://hub.docker.com>
- MariaDB Docker Image https://hub.docker.com/_/mariadb
- GitHub for version control <https://github.com>