Database Selection
Based on the Django documentation, there are 5 officially supported databases that can be implemented easily into Django. Apart from this, there are a number of other third-party database services that can be supported within Django; however, they may require extra work in order to implement.

For the sake of simplicity for this project, we will evaluate and review the following 5 officially supported databases and compare their merits and deficits to each other:
- PostGreSQL
- MariaDB
- MySQL
- Oracle
- SQLite

We will be evaluating each of these in regards to the following:
- Cost of the software
- Ease-of-use/Complexity
- How well it fits our use-case (storing image data)
- Popularity/Employability

PostGreSQL
+ Good for large and complex queries, to store and analyze data
+ Largely Compliant with SQL Standard
+ Possible to process complex data types (geographical data)
+ Flexible full text search
+ Creation of own functions, triggers, data types, etc
+ Most Advanced Open Source DB
+ JSON Support

- Comparatively low reading speed, especially for large bulk operations or read queries
- Documentation can be spotty
- Configuration can be confusing

MariaDB
+ Improved version of MySQL
+ Open Source & Free
+ Good for large data sets
+ Backwards Compatible
+ Fast and reliable

- Not fully compatible with MySQL
- 14th most popular DB (1.95% market share)
- Amazon Aurora can be much faster than MariaDB if exercising AQS
- Expensive Support System
- Not suitable for semi-structure data (JSON)

MySQL
+ Good for simple operations, read & write. Simple data transactions
+ Free
+ Transactions can be rolled back
+ Uses views, triggers, and stored procedures
+ Scalable to handle more than 50mil+ rows, default file size limit is 4gb.

- Not good with large DB
- Spend lots of time to get MySQL to do things that other DBs do automatically
- No support for XML
- No support
- Not as good debugging compared to paid DB
- Prone to data corruption as its inefficient with handling transactions
- Does not support SQL check constraints.

Oracle
+ Cutting Edge
+ Robust Tooling
+ Advanced Multi-Model Databases

- Costly
- Significant resources once installed

SQLite
+ Ultra light-weight
+ No installation needed
+ Reliable
+ Portable
+ Accessible
+ Free + Open Source
+ Can be great for most websites (<100k hits/day)

- Database size restricted to 2GB
- Used to handle low-to-medium traffic HTTP requests

MongoDB
+ Fast and Easy to Use
+ Open Source & Free
+ Supports JSON and other NoSQL Documents
+ Data of any structure can be stored
+ Schema can be written without downtime

- SQL is not used as the query language
- Tools to translate SQL to MongoDB queries exist but add an extra step
- Setup can be lengthy
- Default settings are not secure

For the reasons above, the two top-ranked databased are PostGreSQL and **SQLite**.

For simplicity sake, SQLite will be easier to implement and coordinate on, as it will be hosted alongside our server. There will be a small amount of read/write operations which should be suitable for SQLite. Additionally, our max data will not be that large, and we will still use SQL to query SQLite.

PostGreSQL would be a good second option, but it requires a steeper learning curve as it is a more advanced piece of software. It might be worthwhile to consider transferring to PostGreSQL in the future if we need.