# Firebase Upload Testing

**Driver:**

The app is launched and lands at the home screen. Once the dropdown menu is clicked, the main ViewController code is executed (this is where uploadImages() is written)

**SUT:**

The system under test is Firebase's Storage service, specifically the upload of data.

**Stub:**

User never handles this function, so we can simply hard code a test

(e.g. uploadImages(filename: "test_image", num_images: SIZE, data: metaData)

And verify on the Firebase website itself if a file was uploaded.

**Testing Fixture:**

Storing one, known, box cat animation onto Firebase as a series of five images, each around 4KB.

**Setup Phase:**

Being a backend operation, this function runs on all iPhone models listed on XCode and executes along with the main view controller. It is meant to be run only once, when new cats are needed.

**Exercise Phase:**

The application code is built and ran through XCode's iOS simulator. Once inside the application, the home drop-down menu is clicked, triggering a new view to be loaded. This view contains the plane in which all cats will pop into existence, and so this is where the exercise will take place.

**Verify Phase:**

Upon execution, Firebase should successfully read the request and transfer the image data to Storage as a JPEG, not withholding its metadata. There exists a Firebase GUI where Storage data can be inspected and, for this test, the GUI was utilized to verify a successful upload.

**Example**:

- uploadImages(filename: "box", num_images: 5, data: metaData)
- Calls for Firebase Storage to upload **five** cat images with a root filename **box**, the result of which will be:
- box_1.jpg, box_2.jpg, box_3.jpg, box_4, box_5.jpg on Firebase Storage

**Teardown Phase:**

The application is terminated.

# Firebase Download Testing

**Driver:**

The app is launched and lands at the home screen. At this point, the cat view has yet to be loaded, and thus the connection to firebase has been established but not used. Once the dropdown menu is clicked, cat images are downloaded from Firebase.

**SUT:**

The system under test is Firebase's Storage service, used to contain the Piskel cats.

**Stub:**

A call to Firebase Storage returns a StorageReference that, in turn, must be converted to a Swift UIImage for animation and other image manipulation.

**Testing Fixture:**

Loading one, known, box cat animation from Firebase and storing it as an Array<UIImage>() in Swift.

**Setup Phase:**

Being a backend operation, this function runs on all iPhone models listed on XCode and executes along with the main view controller. It is meant to be run only once per view load.

**Exercise Phase:**

The application code is built and ran through XCode's iOS simulator. Once inside the application, the home drop-down menu is clicked, triggering a new view to be loaded. This view contains the plane in which all cats will pop into existence, and so this is where the exercise will take place.

**Verify Phase:**

Upon execution, Firebase should successfully read the request and transfer the image data from Storage as a StorageReference, not withholding its metadata. This reference can then be converted to a UIImage and be ready for animation. For the purposes of this test, all that is necessary is verification that an image was received. To do this, we can create a subview in the main view controller and display the contents of our new data.

**Example**:

- loadCat(filename: "box", num_images: 5)
- Calls for Firebase Storage to download **five** cat images with a root filename **box**, the result of which will be:
- [box_1, box_2, box_3, box_4, box_5] as UIImages

**Teardown Phase:**

The application is terminated.