University of Piraeus

Department of Digital Systems

Postgraduate Program ‘Information Systems & Services’

Τεχνικές μηχανικής μάθησης ενσωματωμένες σε ένα πληροφοριακό σύστημα για την ανάλυση δεδομένων τροφίμων από κοινωνικά δίκτυα

Machine learning techniques integrated in an information system towards the analysis of social media food data

ME1903
Loukas Leppidos Giannios

Supervising Professor: Dimosthenis Kiriazis

PIRAEUS 2021

Abstract

The aim of this paper is to present a complete Mobile application using the Dart and Flutter languages for the User Interface. The application uses Firebase for Authentication, Fire Store for data storage, Firebase Storage for storing images and videos as well as Firebase Cloud Functions. The application is of the Social Media type, i.e. users are allowed to upload videos or pictures, create friends and followers who will have access to the user's profile. In addition, it contains recipes that the user will be able to access through searching through the application. Finally, the application will be able to recognize the content of a user’s food image and suggest corresponding recipes through machine learning.

Table of Contents

1. Introduction

Mobile applications are becoming more and more present in our daily lives, allowing users to perform many tasks through the use of mobile devices. As of November 2016, there is more network traffic from mobile devices (48.19%) compared to desktops or laptops (47%). There are different mobile operating systems with different programming languages and tools.

To be distributed to most users, a mobile application must be adapted to two separate platforms, namely Android and IOS. Obviously, the differences between the two are large and often require different skill sets to develop, such as Java / Kotlin for Android only and Object-C / Swift for iOS only. Thus, developers and companies often struggle to cope with the complexity of developing multi-platform applications.

An application is usually developed with a software development kit (SDK) provided by the mobile operating system developer. Applications can also be developed using cross-platform development approaches. This approach allows developers to use a code base that can be run on multiple platforms. Growth costs are often reduced and offer other benefits such as faster growth and release of overhead costs.

Flutter is a new multi-platform technology that promises high performance applications. Provides widgets for Android and iOS thus offering high user satisfaction. This study explores the development of a Flutter application. The programming language behind Flutter is Dart and a combination of these will be used in this work to develop the application. The Dart programming language was developed by Google and can also be used to build server and desktop applications.

Machine Learning is a science that helps applications to offer many features to its users. Most common applications such as Facebook, Instagram, Twitter and many more use machine learning in different parts of the applications. Machine Learning is described with details in the next chapter.

The power of social media is growing enormously and developing a social media app is a hot trend these days. The thesis aimed to create a social media application with an aspect of Machine Learning for general purpose goal accomplishment. Its purpose is to explore how to design and create a social media application to improve the users intent to achieve their goals.

1.1 Social Media Need

Millions of people use social media to connect, meet and share. Online social networks are configuration of relationships to facilitate social interactions which are offered by new web or Mobile applications, ranging communications among people from casual friendships to professional networks found in businesses. By the advent of the Internet, geographical limits are disappeared by diversified social ties whereas people are enabled to connect to the world through making new connections and using specific services of social media applications. In fact, the new virtual communities build on social media enable people to form globalized relationships [1].

In most of social media platforms, users are easily attracted to the open-networking because of their friends and/or their peers' membership. Each individual can lose or refuse relationships/connections when find them no more useful, an action which is less frequent in the real world. Users of social media are able to make unlimited friendships through joining social groups or adding someone to their friends or connection list.

Almost all of social units, professionals, organizations and groups of traditional society have noticed the power of social media and are aware of the values added to their activities by joining this phenomenon.

1.2 Social Media Importance

Today there exist a plethora of social media applications. These usually provide a specific set of services to their users. The most apparent example is Facebook, which is a social network for everybody. Some others are Linked In, Twitter and Instagram. However, do these social networks help the users themselves, or are they just a waste of time? Regardless of the answer, people often have goals they want to achieve. Yet, distraction is everywhere and without a solid habit, it is easy to forget about a personal goal. Ironically distractions quite often come in the form of notifications from social media, or various push notifications of minor importance [2].

The important role of social media for scientific research becomes evident by the sheer number of papers analyzing data from social media platforms [3]. Various review studies and meta-analyses provide an overview of how data extracted from different social media platforms are analyzed, and how social media apps are used in different contexts and environments. For example, one paper reviews literature on the use of social media in academia [4]. It distinguishes between several categories of social media use, including social networking, social data sharing, video, blogging, wikis, rating, and reviewing. It reports that the percentage of scholars who use social networking apps (Facebook, LinkedIn) for professional purposes is much lower than the percentage of scholars who use it for personal reasons. It also points towards a large variability in usage of different platforms among scholars, with numbers ranging between 10% for Twitter, 46% for ResearchGate, and 55% for YouTube. Another systematic review analyzes social media use for public health communication among the general public, patients, and health professionals based on 98 original research studies [5]. These studies included a range of social media tools and apps, with Facebook, blogs, Twitter, and YouTube being the most often reported tools.

1.3 Related Works

As mentioned before, some examples of worldwide known social media apps are Linked In, Twitter and Instagram and of course Facebook. All of these apps are using Machine Learning techniques to enhance users experience.

1.3.1 Facebook

Launched in a Harvard dormitory in 2004 by Mark Zuckerberg, Eduardo Saverin, Chris Hughes, and Dustin Moskovitz, the project initially had high school and college students in mind. It attracted seven million users in its first two years of existence. By February 2010 Facebook was reported to have more than 400 million active members [7].

Facebook users can create profiles with photos, lists of personal interests, contact information, and other personal information. Communicating with friends and other users can be done through private or public messages or a chat feature. Also, users can create and join personal interest and fan groups, some of which are maintained by organizations as a means of advertising. Unlike other social network sites, Facebook clearly distinguishes between brands and regular members. Instead of profiles, brands establish communities using Facebook pages.

For researchers, Facebook constitutes a rich site for those interested in studying the phenomena of social networks due to its heavy usage patterns and technological capacities that bridge online and offline connections.

Facebook is at the top of the social media game as its platform caters to a wide variety of people, incorporating many different media aspects, from photos to messenger to text. It is not as limited as LinkedIn and Twitter, which typically cater to a specific demographic. Because of its wide appeal, Facebook has attracted a significant user base, which translates to ad revenue, since companies desire to spend their ad budgets on platforms that receive the most viewership, and with 2.74 billion active users monthly, it's hard to top Facebook [8].

When users are about to tag someone on Facebook, before even mentioning the name of the person in the image, Facebook gives them a suggestion and 99.99% it gives the right name. How does Facebook know the name of the person users are about to tag in the image? The credit goes to machine learning. This is just a small part of big data machine learning. AI and machine learning give the exact insights based on the data collected.

The most controversial technology of the Facebook AI suite is the famous Deep Face network for face recognition. A nine-layer neural network with over 120 million connection weights that’s been trained with billions of data samples, it tracks digitized human images and makes the search of a person by a single photo very fast and accurate.

There are many more parts that Facebook uses machine learning such as targeting advertising, language translation, news feed and more.

1.3.2 Instagram

Instagram is a photo sharing application (Instagram 2014) in which the user can share the world through their eyes to showcase what they find interesting or important. The majority of pictures range from mundane events, such as what the user is eating, reading or listening to, to the events attended by the user; parties, concerts, get together, etc. Instagram allows users to edit and upload photos and short videos through a mobile app. Users can add a caption to each of their posts and use hashtags and location-based tags to index these posts and make them searchable by other users within the app. Each post by a user appears on their followers Instagram feeds and can also be viewed by the public when tagged using hashtags or tags. Users also have the option of making their profile private so that only their followers can view their posts [9].

As with other social networking platforms, Instagram users can like, comment on and bookmark others posts, as well as send private messages to their friends via the Instagram Direct feature. Photos can be shared on one or several other social media sites including Twitter, Facebook and Tumblr with a single click.

Instagram is not only a tool for individuals, but also for businesses. The photo-sharing app offers companies the opportunity to start a free business account to promote their brand and products. Companies with business accounts have access to free engagement and impression metrics. According to Instagram's website, more than 1 million advertisers worldwide use Instagram to share their stories and drive business results. Additionally, 60% of people say they discover new products through the app. In addition, Instagram provides a wide range of digital filters that can be applied to users' photos [10].

Instagram uses machine learning as well. Some examples are described below:

- *Target advertising:* In order to make the data that Instagram collects valuable; it must extract customer insights from it. By assessing the search preferences and engagement insights from its users, Instagram can sell advertising to companies who want to reach that particular customer profile and who might be most interested in receiving a particular marketing message. Since Facebook with 1.8 billion users owns Instagram they have a powerful network of analytics information to help target advertising based on what people like, who they follow and interact with and what they save.
- *Enhance the User Experience:* In order to ensure users find value in the platform, it's important for Instagram to show them what they will like. As the amount of content grows, finding content that each user will find relevant becomes exponentially more challenging. When Instagram changed its feed from reverse-chronological order to showing posts that they believe users would like and share, machine-learning algorithms were put on the job to help sort the information and to better learn over time what is most valued and relevant for each user to create a personalized feed.
- *Filter spam:* Instagram uses artificial intelligence to fight spam. The spam filter is able to remove fake messages from accounts written in nine languages including English, Chinese, Russian, Arabic and more. Once messages are detected they are automatically removed. Instagram uses Facebook's artificial intelligence algorithm Deep Text that is able to understand the context of a message nearly as good as humans.
- *Fight Cyberbullying and Delete Offensive Comments:* In a survey conducted by Ditch the Label, 42% of more than 10,000 UK youth between ages 12 and 25 reported Instagram was the platform where they were most bullied. With this unfortunate distinction of having the biggest cyberbullying problem of any social media site, they became the first to use machine learning to automatically remove offensive posts, whereas Facebook and Twitter rely on users to report abusive language. Based on the success of using Deep Text to identify spam and remove it, Instagram officials began to see it as a solution to identify and eliminate comments that violate Instagram's Community Guidelines. Humans reviewed and tagged actual Instagram posts to help Deep Text learn what would be considered offensive content in certain contexts and what wouldn't be. If the algorithm finds something offensive, it is immediately removed. Even though Deep Text isn't perfect, it has received praise from cyberbully prevention organizations for the work it is doing. There is still the risk of misclassifying something as offensive when it's not or that offensive comments will make it through.

1.3.3 Twitter

Twitter has become increasingly popular with academics as well as students, policymakers, politicians and the general public. Many users struggled to understand what Twitter is and how they could use it, but it has now become the social media platform of choice for many.

Twitter is a 'micro-blogging' system that allows you to send and receive short posts called tweets. Tweets can be up to 140 characters long and can include links to relevant websites and resources. Twitter users follow other users. If users follow someone, they can see their tweets in their twitter 'timeline' [11]. Users can choose to follow people and organizations with similar academic and personal interests to them. Users can create their own tweets or they can retweet information that has been tweeted by others. Retweeting means that information can be shared quickly and efficiently with a large number of people [12].

Twitter allows you to:

- easily promote your research, for example by providing links to your blog stories, journal articles and news items.
- reach a large number of people quickly through tweets and retweets.
- follow the work of other experts in their field.
- build relationships with experts and other followers.
- keep up-to-date with the latest news and developments, and share it with others instantly.
- reach new audiences.
- seek feedback about your work and give feedback to others.
- follow and contribute to discussions on events, for example conferences that you can't attend in person.
- express who you are as a person.

1.3.4 LinkedIn

LinkedIn is the world's largest professional network on the internet. Users can use LinkedIn to find the right job or internship, connect and strengthen professional relationships, and learn the skills they need to succeed in their career. Users can access LinkedIn from a desktop, LinkedIn mobile app, mobile web experience, or the LinkedIn Lite Android mobile app [13].

A complete LinkedIn profile can help users connect with opportunities by showcasing their unique professional story through experience, skills, and education. Users can also use LinkedIn to organize offline events, join groups, write articles, post photos and videos, and more.

LinkedIn is a platform for anyone who is looking to advance their career. This can include people from various professional backgrounds, such as small business owners, students, and job seekers. LinkedIn members can use LinkedIn to tap into a network of professionals, companies, and groups within and beyond their industry [14].

2. Machine Learning

Machine learning (ML) is the study of computer algorithms that are automatically enhanced by experience and the use of data. It is considered part of artificial intelligence. Machine learning algorithms create a model based on some data samples, known as "training data", in order to make predictions or decisions without being programmed to do so. Such algorithms are used in a wide variety of applications, such as email filtering and computer learning machines that focus on the use of data and algorithms to mimic the way people learn, gradually improving their accuracy.

Machine learning is one of the many fields of artificial intelligence, about how computers learn from experience to improve their ability to think, plan, decide and act. There are three different types machine learning [15]. These are Supervised Learning, Unsupervised Learning and Reinforcement Learning as shown in Figure 2.1.

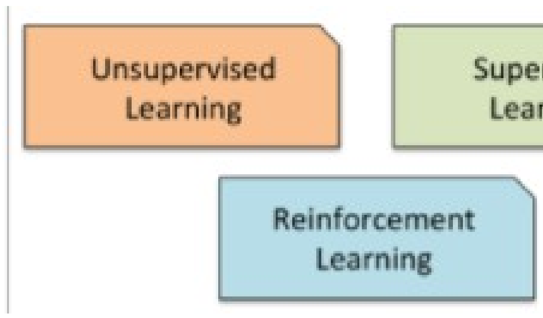


Figure 2.1 Machine Learning Types

- Supervised learning: Examples of inputs and desired outputs are given on the computer, given by a "teacher", and the goal is to learn a general rule that maps inputs to outputs.
- Unsupervised learning: No data is given to the learning algorithm, leaving it alone to find structure in its input. Unsupervised learning can in itself be a goal (discovering hidden patterns in data) or a means to an end (learning characteristics).
- Reinforcement learning: A computer program interacts with a dynamic environment in which a specific goal must be achieved (such as driving a vehicle), without a teacher explicitly telling him if he has come close to his goal. Another example is learning to play a game against an opponent.

Another categorization of machine learning problems is divided into the following:

- Classification, the input data is divided into two or more classes, and the machine must construct a model, which will assign the data to one or more (multi-label classification) classes.
- Regression, also a problem of supervised learning, the results are continuous and not discrete.
- Clustering, a set of inputs is going to be divided into groups. Unlike classification, groups are not known in advance, making this separation a typical non-supervised learning task.
- Density estimation, where it finds the distribution of input data in a space.
- Dimensionality reduction problems, where data is simplified and mapped to a smaller space. Topic modeling is a related problem, where the machine is called upon to find documents that cover similar topics from a set of documents written in natural language.

2.1 Image Processing

Computer vision is important subject in image processing. General description of computer vision in image processing can be summarized in following steps:

- Image capture - Image is captured (by camera or similar device) and digitized.
- Pre-processing - Digitized image is modified to emphasize important features (e.g. noise reduction, contrast normalization etc.).
- Segmentation - Selection of interesting features (edges, similar surfaces).
- Description - Extraction of radiometric, photometric descriptors and so on.
- Classification - Some means to classify the object.

Even though machine learning existed as field of study since the second half of the 20th century, there was no wider adoption of its techniques in image processing for a very long time. It was first introduced in the classification step of the processing pipeline. In other words, complex problems were simplified by reducing the visual information contained within the image into handful of simple features that were fed into machine learning model. This approach consequently favors very simple models such as K-nearest Neighbors (KNN) or Support Vector Machines (SVM).

This brings the problem that these applications are not very versatile. Each application is usually only capable of solving very narrow problem and any deviation from ideal circumstances can mean failure. Application can have problems with varying contrast, illumination, scaling, rotation etc.

Second problem is often the fact that because image has to be pre-processed several times before it is fed into machine learning model it requires extra time and computation resources. This is less of a problem with current hardware innovations, but it is still not negligible factor and it can have negative effect on the cost of the solution. This is where machine learning in general indicates significant advantage.

Common approach to computer vision can find success in applications that are deployed in very restricted environments with rigid constraints. In controlled environment, it is usually very simple to describe the problem in formal rules. Even though it can be still found in certain places, it starts to be forced out by application of machine learning simply because the barrier of entry for wider adoption decreases every day [16].

2.2 Unsupervised Learning

In this learning approach, the model is training by observing new data and extracting patterns in the date without being instructed on what they are. Opposed to supervised learning described, the advantage of this approach is that the model is able to learn from data without supervision (as the name suggests). This means that there is no need for input data to be annotated, therefore it takes much less time and resources to deploy these models in practice. The biggest hurdle of supervised learning approach in real world applications is to obtain appropriate data. Appropriate data in this context means, data that were somehow classified into different categories, which can be very tedious and slow process. In some cases, the task itself prevents the usage of labeled data.

Majority of unsupervised learning algorithms belong to group called clustering algorithms. These algorithms are centered on the idea to analyze geometric clustering of data in input space to determine their affiliation. This is achieved by the presupposition that data point clustering in input space are likely to exhibit similar properties [17].

Examples of unsupervised learning models are:

• K-means - clustering model.

• Self Organizing Maps (SOMs) - instance based.

• Principal Component Analysis (PCA) - dimensionality reduction.

2.3 Supervised Learning

Supervised learning approach is more commonly used. This approach requires training data with specific format. Each instance has to have assigned label. These labels provide supervision for the learning algorithm. Training process of supervised learning is based on the following principle. Firstly, the training data are fed into the model to produce prediction of output. This prediction is compared to the assigned label of the training data in order to estimate model error. Based on this error the learning algorithm adjusts model's parameters in order to reduce it [18].

2.4 Algorithms Structure

Although machine learning algorithms are diverse and are using different techniques its structure can be generalized. Structure of nearly all machine learning algorithms can be described as composition of the following components:

• Dataset specification

• Model

• Cost function

• Optimization procedure

Almost all supervised learning algorithms use the same Dataset specification. The other three components can vary dramatically. This level of analysis is useful for building of intuition for Neural Networks (NNs) and explanation of its individual components.

2.4.1 Dataset Specification

Supervised learning requires datasets with specific properties. Each dataset contains set of n instances which consists of a pair of input vector xi and output scalar yi.

Individual components of input vector have to be of unified type. In case of input data in form of image it is value for individual pixels. In other cases, they can be real values. Almost universally in machine learning it stands that input should be normalized. This presumption holds in images automatically since each pixel has to have its vales in fixed range. It is very important in other types of machine learning tasks, where this is not guaranteed.

Output scalar yi represents class of given instance. Type of this output value thus has to acquire only certain values. To put it differently, it has to be a set of cardinalities equal to number of all possible classes.

2.4.2 Model

Model is prediction apparatus that takes input xi to predict value of its output yi. Each model has parameters represented by vector θ, which are adjusted during the training process. The simplest example of model type is linear model, also called linear regression.

2.4.3 Cost Function

To achieve the learning ability of the machine learning algorithm, it is necessary to estimate the error of its predictions. This is estimated with so called cost function (also sometimes called loss function).

This function has to have certain properties. Ability of the machine learning algorithm to learn rests on the estimation of its improvement with change of its parameters. Therefore, cost function has to be at least partially differentiable. In case of linear regression, it is most common to use sum of square error. The main reason being that derivative of this function for linear model has only one global minimum.

2.4.4 Optimization Procedure

The last part of learning algorithm is the optimization procedure. It consists of update of model's parameters θ in order to improve its prediction. In other words, to find θ such that the value of cost function $J(\theta)$ for given dataset is as small as possible. Unfortunately, only very simple problems can be approximated using model as simple as linear regression. More complex model usually means more complicated cost function. Optimization process of more complex cost functions cannot be guaranteed to find global minimum. In this case, the optimization procedure has to be of iterative character. To put it in a different way, algorithm has to approach the minimum of iterations. Many of the iterative methods belong to the group called gradient based optimization.

2.5 Model Complexity

In the first approximation it could be said that the task of supervised machine learning is to model relationship between the input output data most accurately. The problem with this definition is that in the practical application there is never enough data to capture true relationship between the two. Therefore, the task of machine learning is the attempt to infer true relationship by observing incomplete picture.

The most important property of machine learning model is its generalization ability. That is ability to produce meaningful results from data that were not previously observed.

Generalization ability is dependent on complexity of the model and its relationship to complexity of underling problem. When model does not capture complexity of the problem sufficiently it is described as under fitting. In case the complexity of model exceeds the complexity of underling problem then this phenomenon is called over fitting.

In both of these extremes the generalization ability suffers. In the former case the model is unable to capture true intricacies of the problem and therefore is unable to predict desired output reliably. In the latter case it tries to capture even the subtlest data perturbation that might be in fact a result of stochastic nature of the problem and not the real underlying relationship. This can also cause the fact that input data is missing some variable necessary to capture the true relationship. This fact is unavoidable and it thus has to be taken into account when designing machine learning model. Depiction of this phenomena in case of two variable inputs is on Figure 2.2.

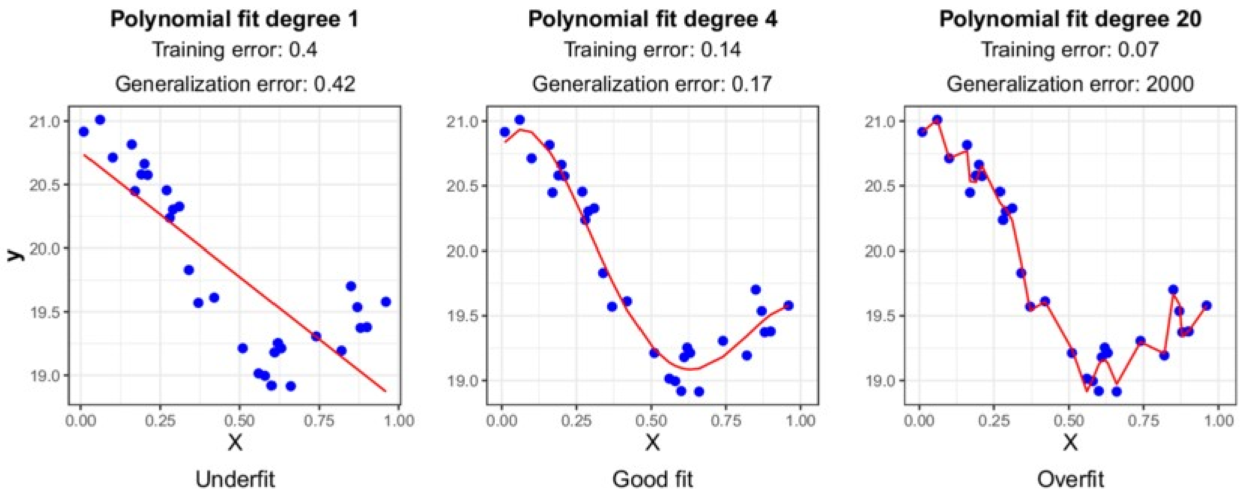


Figure 2.2 Levels of Generalization

Typically, the machine learning model is trained on as much input data as possible in order to achieve the best possible performance. At the same time its error rate has to be verified on independent input data to check whether the generalization ability is not deteriorating. This is typically achieved by splitting available input data into training and testing set (usually in 4:1 ratio for training to test data). Model is trained with training data only and the performance of the model is tested on the test data. Even though the true generalization error can never be truly observed, its approximation by test error rate is sufficient for majority of machine learning tasks.

Regularization is any modification that is made to the learning algorithm that is intended to reduce its generalization error but not its training error.

As it has already been mentioned, the most important aspect of machine learning is striking the balance between over and under fitting of the model. To help with this problem concept of regularization was devised. It is a technique that helps to penalize the model for its complexity. Basic concept consists of adding a term in the cost function that increases with model complexity [19].

2.6 Convolutional Neural Networks

CNNs are specialized type of neural networks that was originally used in image processing applications. They are arguably most successful models in AI inspired in biology.

Even though they were guided by many different fields, the core design principles were drawn from neuroscience. Since their success in image processing, they were also very successfully deployed in natural language and video processing applications.

Aforementioned inspiration in biology was based on scientific work of David Hubel and Torsten Wiesel. Neurophysiologists Hubel and Wisel, investigated vision system of mammals from late 1950 for several years. In the experiment, that might be considered little gruesome for today's standards, they connected electrodes into brain of anesthetized cat and measured brain response to visual stimuli. They discovered that reaction of neurons in visual cortex was triggered by very narrow line of light shined under specific angle on projection screen for cat to see. They determined that individual neurons from visual cortex are reacting only to very specific patterns in input image. Hubel and Wiesel were awarded the Nobel Prize in Physiology and Medicine in 1981 for their discovery [20].

2.6.1 Convolutional Neural Network Structure

Structure of Convolutional networks is typically composed of three different types of layers. Layer can be either Convolutional, Pooling or fully connected. Each type of layer has different rules for forward and error backward signal propagation.

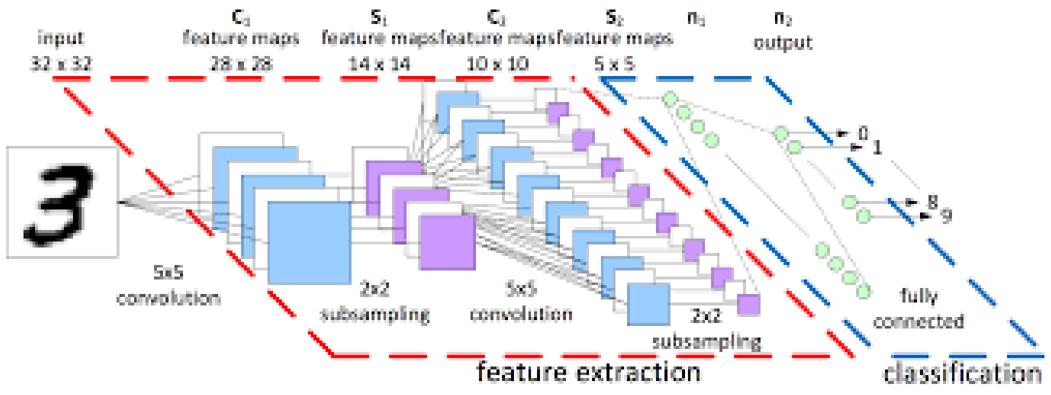


Figure 2.3 CNN Structure

There are no precise rules on how the structure of individual layers should be organized. However, with exception of recent development6 CNNs are typically structured in two parts. First part, usually called feature extraction, is using combinations of convolutional and pooling layers. Second part called classification is using fully connected layers. This is illustrated in Figure 2.3.

2.6.1.1 Convolutional Layer

The convolutional layer employs convolution operation. Input into this layer is simply called input. Convolution operation is performed on input with specific filter, which is called kernel. Output of convolution operation is typically called feature map.

Input into Convolutional layer is either image (in case of first network layer) or feature map from previous layer. Kernel is typically of square shape and its width can range from 3 to N pixels. Feature map is created by convolution of kernel over each specified element of input. Convolution is described in more detail in section describing training of CNN.

Depending on the size of kernel and layer's padding preferences the process of convolution can produce feature map of different size than input. When the size of output should be preserved it is necessary to employ zero padding on the edges of input. Zero padding in this case has to add necessary amount of zero elements around the edges of input.

Reduction of feature map can go even further in case of use of stride. Application of stride specifies by how many input points is traversed when moving to neighboring position in each step. When the stride is 1, kernel is moved by 1 on each step and the resulting size of feature map is not affected.

Each Convolutional layer is typically composition of several different kernels. In other words, output of this layer is tensor containing feature map for each used kernel. Each of these is designed to underline different features of input image. In the first layers these features are typically edges. Higher the layer, the more complex features are captured.

Each kernel that is used is applied to all inputs of the image to produce one feature map which basically means that neighboring layers are sharing the same weights. This might not be sufficient in some applications and therefore it is possible to use two other types of connections. Locally connected which basically means that applied kernel is of the same size as the input and tiled convolution which means alternation of more than one set of weights on entire input.

2.6.1.2 Pooling Layer

The pooling layer doesn't constitute any learning process but it is used to down-sample size of the input. The principle is that input is divided into multiple not overlapping rectangular elements and units within each element are used to create single unit of output. This decreases the size of output layer while preserving the most important information contained in input layer. In other words, pooling layer compresses information contained within input.

Type of operation that is performed on each element determines a type of pooling layer. This operation can be averaging over units within element, selecting maximal value from element or alternatively learned linear combination of units within element. Learned linear combination introduces form of learning into the pooling layer, but it is not very prevalent.

Selecting of maximal value is most common type of pooling operation and in that case the layer is called Max-Pooling accordingly. Positive effect of Max-pooling down-sampling is that extracted features that are learned in convolution are invariant to small shift of input.

2.6.2 Convolutional Neural Network Training

Optimization process of CNN is complicated because network is composed of different types of layers. Forward signal propagation and backward error propagation are following special rules for each layer.

First phase is called forward-propagation, where the signal is propagated from inputs of the CNNs to its output. In the last layer the output is compared with desired value by cost function and error is estimated. In second phase is again used back-propagation algorithm to estimate error contribution of individual units. Variable parameters of the network are again optimization by gradient descent algorithm.

2.6.3 Regularization

Control of complexity applies to both Neural Networks and CNN. There are several popular regularization techniques that mostly consist of modification of cost function or optimization algorithm. Slightly different approach is to modify structure of the network during training phase.

The best regularization method is to combine predictions of many different models. This method greatly improves generalization ability of combined model while preventing over-fitting. Exactly on this idea are based ensemble models. The problem with ensemble models is that they are computationally expensive. Because of this, ensembles are usually composed of many very simple models.

This idea is especially problematic with Deep Neural Networks, which are model with many parameters that are difficult to train. Moreover, even when trained models are available in some applications it still isn't feasible to evaluate many different models in production environment. Another problem is that there might not be enough data to train these different models.

All of these problems can be solved by dropout technique. The basic idea is that each neuron in the network has certain probability to be deactivated during one iteration. This potential for deactivation is evaluated in every iteration, to ensure that network has different architecture every time. Deactivated means that it will not propagate any signal through. This forces individual neurons to learn features that are less dependent on its surrounding.

Probability for deactivation is a hyper-parameter that can be tuned, but reasonable default value is 0.5. Dropping out is only happening in the training phase. In testing phase are all weight connection multiplied by the probability of a dropout. This is done because the activation of the network has to stay roughly equivalent7 in both training and testing phase.

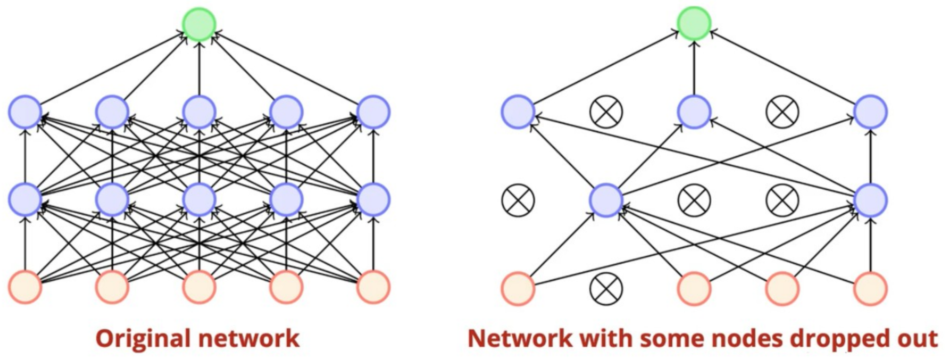


Figure 2.4 Network before and after Dropout

2.7 Convolutional Neural Network Tools

In order to select appropriate tool for implementation of CNN for classification, investigation of available software tools and libraries was conducted. There is vast variety of software tools for machine learning. Some of these are general tools for machine learning, but some are specifically designed for deep learning.

In the last 10, years the software tools for machine learning have undergone a renaissance. There is broad selection of them available and new tools are introduced quite frequently. Almost every commonly used programming language has either some software library or at least some available Application Programming Interface (API).

The selection of the software tool was influenced by several factors. Firstly, the implementing language had to be well known and somewhat mainstream. Enough of available learning materials had to be available, preferably in form of tutorials. The most important factor was good support for learning on GPU.

The different software tools are described below:

- **Theano**: It is a python library. Designed to define, optimize and evaluate mathematical expression with multi-dimensional arrays. This makes it suitable for machine learning needs. Theano is built on top of NumPy, which is python module that enables efficient operation with tensors and basic image processing technique. Combination of NumPy and SciPy brings rich set of tools for image processing and data processing. Its capabilities can arguably rival Matlab, while being open source and free. Theano's biggest rival is currently TensorFlow project. One of the problems of Theano is its low-level nature. Implementation of machine learning algorithms directly can be very complicated. This is probably the reason it slowly falling by the way side. This is also the reason why Theano as a tool is not very suitable for direct implementation of CNN models [21].
- **Torch**: Torch is one of the oldest frameworks for scientific computing. It is rich and powerful tool that was one the first heavily used for deep learning applications. Similarly to other items in this list it offers fast and efficient GPU support. Minor negative of Torch is that it uses Lua scripting language as a programming interface. Lua is not very commonly used and as such it suffers from lack of interest of the mainstream machine learning community [22].

- **TensorFlow**: TensorFlow is very similar to Theano. As the name suggest this library is focused on effective work with tensors. It was originally developed for internal use in Google for machine learning task but it was released as open source in 2015. TensorFlow's computations are expressed as stateful dataflow graphs, which enables efficient support for GPU aided computation. Is currently advertised as one of the fastest frameworks for deep learning needs. Its disadvantage is similar to Theano, in the fact that it is very low level and direct usage for implementation of Deep learning models is not ideal [21].
- **Caffe**: Caffe is a deep learning framework that aims to be modular and fast. It is developed by Berkeley AI Research (BAIR) and by community contributors. It is implemented in C++ but it also offers APIs for several other languages as for example python. Its biggest drawback is its lack of quality documentation. This fact is partially remedied by the existence of Model Zoo, which is collection of favorite models that are freely available. Caffe was in the last years used by companies as Facebook for example mainly because its performance capabilities. Caffe is more geared towards the development of mass production application than it is for research purposes [22].
- **Keras**: Keras is relatively young but mature project written in python. It is high lever neural network API. It is built capable of running on top of either Theano or TensorFlow libraries. It is very simple with emphasis on rapid model development. At the same time thanks to python infrastructure, it is very easily extensible. Keras probably currently has one of the largest communities among similar tools for deep learning. It has very good documentation containing many code examples and other resources that help users to get started very quickly.

For the scope of this thesis, we used the combination of Keras and TensorFlow to build the model we needed for an image recognition feature in our application. The main reason of this choice was the large documentation that help the studying process and the fact that it can be used in python, a programming language we are familiar with.

3. Image Processing Implementation

Our main goal was to add an image recognition feature to our application that lets users take a picture of their food and similar recipes of the recognized food will be recommended to them. In order to do that, we used a dataset called Food-101 and trained a model to recognize an input image.

3.1 About The Dataset

This dataset consists of 101 food categories, with 101.000 images, a thousand for each category. For each class, 250 manually reviewed test images are provided as well as 750 training images. On purpose, the training images were not cleaned, and thus still contain some amount of noise. This comes mostly in the form of intense colors and sometimes wrong labels. All images were rescaled to have a maximum side length of 512 pixels.

The dataset is available to the public by Kaggle, an online community that offers a public data platform, a cloud-based workbench for data science, and Artificial Intelligence education.

The dataset structure is shown below:

- meta folder that contains the text files - train.txt and test.txt
- train.txt that contains the list of images that belong to training set
- Test.txt that contains the list of images that belong to test set
- Classes.txt that contains the list of all classes of food
- images folder that contains 101 folders with 1000 images each.

3.2 Colab Notebook

Machine learning code execution, as well as the algorithms used, often lead to a time-consuming and quite difficult process for ordinary desktops or laptops. For this reason, for the purposes of this work, Google Colab was used to train the model.

Google Colab is a free cloud-based Jupyter notebook environment. Therefore, no installation is required while the code files that are created are accessible to anyone we want, for any editing or reading. Another advantage is that it contains the most famous Python libraries installed and especially those used for machine learning (Keras-TensorFlow).

Developers using Colab are able to:

- Write and execute code in Python.
- Create, upload and share files.
- Import and save files to and from Google Drive.
- Import and publish files from GitHub.
- Enter datasets.

Google Account requires a user account to use Colab. For the scope of this thesis, we used Google Colab in order to train our model.

3.3 Model Building Blocks

In order to train a model, we used Inception-v3 algorithm which is included in Keras. Inception-v3 is a convolutional neural network architecture from the Inception family that makes several improvements including using Label Smoothing, factorized 7×7 convolutions, and the use of an auxiliary classifier to propagate label information lower down the network (along with the use of batch normalization for layers in the side head).

3.3.1 Imports

The used libraries in order to train the model, are shown below.

```
import tensorflow as tf
import matplotlib.image as img

import numpy as np

from collections import defaultdict

import collections

from shutil import copy

from shutil import copytree, rmtree

import tensorflow.keras.backend as K

from tensorflow.keras.models import load_model

from tensorflow.keras.preprocessing import image

import matplotlib.pyplot as plt

import numpy as np

import os

import random

import tensorflow as tf

import tensorflow.keras.backend as K

from tensorflow.keras import regularizers

from tensorflow.keras.applications.inception_v3 import InceptionV3

from tensorflow.keras.models import Sequential, Model

from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten

from tensorflow.keras.layers import Convolution2D, MaxPooling2D, ZeroPadding2D, GlobalAveragePooling2D,
AveragePooling2D

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.callbacks import ModelCheckpoint, CSVLogger

from tensorflow.keras.optimizers import SGD

from tensorflow.keras.regularizers import l2

from tensorflow import keras

from tensorflow.keras import models

from keras.layers.normalization import BatchNormalization
```

3.3.2 Variables and ImageDataGenerator

The variables used are the following, where n_classes is the number of classes used which is 101 classes, img_width and img_height is the width and height of the images, train_data_dir and validation_data_dir are the directories of the train and test images folders. The number of train samples is 75750 and the testing samples is 25250.

```
K.clear_session()

n_classes = 101

img_width, img_height = 299, 299

train_data_dir = r'/content/train'

validation_data_dir = r'/content/test'

nb_train_samples = 75750

nb_validation_samples = 25250

batch_size =64
```

During the image processing we used ImageDataGenerator function which generates batches of tensor image data with real-time data augmentation.

```
train_datagen = ImageDataGenerator(

   rescale=1. / 255,

   shear_range=0.2,

   zoom_range=0.2,

   horizontal_flip=True)


test_datagen = ImageDataGenerator(rescale=1. / 255)


train_generator = train_datagen.flow_from_directory(

   train_data_dir,

   target_size=(img_height, img_width),

   batch_size=batch_size,
   class_mode='categorical')



validation_generator = test_datagen.flow_from_directory(
   validation_data_dir,
   target_size=(img_height, img_width),
   batch_size=batch_size,
   class_mode='categorical')
```

3.3.3 Keras Layers

The next part of the process is the Inception-v3 algorithm implementation. The Keras layers are illustrated below:

```
inception = InceptionV3(weights='imagenet', include_top=False , input_shape=(299,299,3))
x = inception.output
x = GlobalAveragePooling2D()(x)
x = Dense(4096)(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)
x = Dropout(0.5)(x)
```

- **GlobalAveragePooling2D** applies average pooling on the spatial dimensions until each spatial dimension is one, and leaves other dimensions unchanged. In this case values are not kept as they are averaged.
- **Dense(Number_of_units)** creates a fully connected layer where Number_of_units is a number of fully connected neurons in one layer and in our case is 4096.
- **Batch-normalization** layer does the following:
    1. Calculates the mean and variance of the layers input.
    2. Normalizes the layer inputs using the previously calculated batch statistics.
    3. Scales and shifts in order to obtain the output of the layer.
- **Activation("relu")** where relu stands for Rectified Linear Unit and is the most commonly used activation function. The function returns 0 if it receives any negative input, but for any positive value x. it returns that value back. So, it can be written as f(x)=max (0, x).
- **Dropout(num)** is an incredibly popular method to combat overfitting in neural networks and num is both probability that any unit is dropped and also the coefficient by which are the outputs multiplied

during forward evaluation, in our case is 0.5 (50%). The idea behind Dropout is to approximate an exponential number of models to combine them and predict the output. In machine learning it has been proven the good performance of combining different models to tackle a problem, or combining models trained in different parts of the dataset.

3.3.4 Model Compilation

When the structure of the model is specified, before it can be trained it also needs to have cost function, optimization procedure and metrics defined. This is done by calling compile method on the model.

```
predictions = Dense(n_classes,kernel_regularizer=regularizers.l2(0.005), activation='softmax')(x)

model = Model(inputs=inception.input, outputs=predictions)

model.compile(optimizer=SGD(lr=0.01, momentum=0.9), loss='categorical_crossentropy', metrics=['accuracy'])
```

Where parameter loss specifies cost function, optimizer optimization procedure and metrics specifies metrics by which the model is measured.

3.3.5 Model Fitting

Very useful property of fit_generator method is that while the model is trained on GPU it is able to prepare another batch of training data in parallel. This is very useful because the data augmentation performed by the generator does not slow down the training process. The model fitting is shown below.

```
history = model.fit_generator(

            train_generator,

            steps_per_epoch = nb_train_samples // batch_size,

            validation_data=validation_generator,

            validation_steps=nb_validation_samples // batch_size,

            epochs=32,

            verbose=1,

           )
```

Method fit_generator has been used to train model with following parameters:

- generator : initialized with train_generator for training data.
- steps_per_epoch : number of calls to train_generator per each epoch.
- epochs : number of epochs.
- validation_data : initialized with validation_generator for testing data.
- validation_steps : number of calls to generate_data per testing.

3.3.6 Model save

The last step of the process was to save the model. This was achieved by the following code:

```
model.save(r'/path/model_trained_3class.hdf5')

tf.keras.models.save_model(model,"/ path /best_model")
```

After saving the model we had to convert the hdf5 file to a type of file that is readable in Flutter. In order to do that, we used a converter called TFLiteConverter that produces a tflite file that can be read in Flutter. The following code illustrates that conversion.

```
import tensorflow as tf

# Convert the model

converter = tf.lite.TFLiteConverter.from_saved_model(r"/path/best_model") # path to the Saved Model directory

tflite_model = converter.convert()


# Save the model

with open(r'/path/best_model/model.tflite', 'wb') as f:

f.write(tflite_model)
```

3.4 Experimental Phase

In this section the process of selection of model structure and its parameters is described and the summarized achieved results on the Food-101 dataset.

3.4.1 Model Parameters

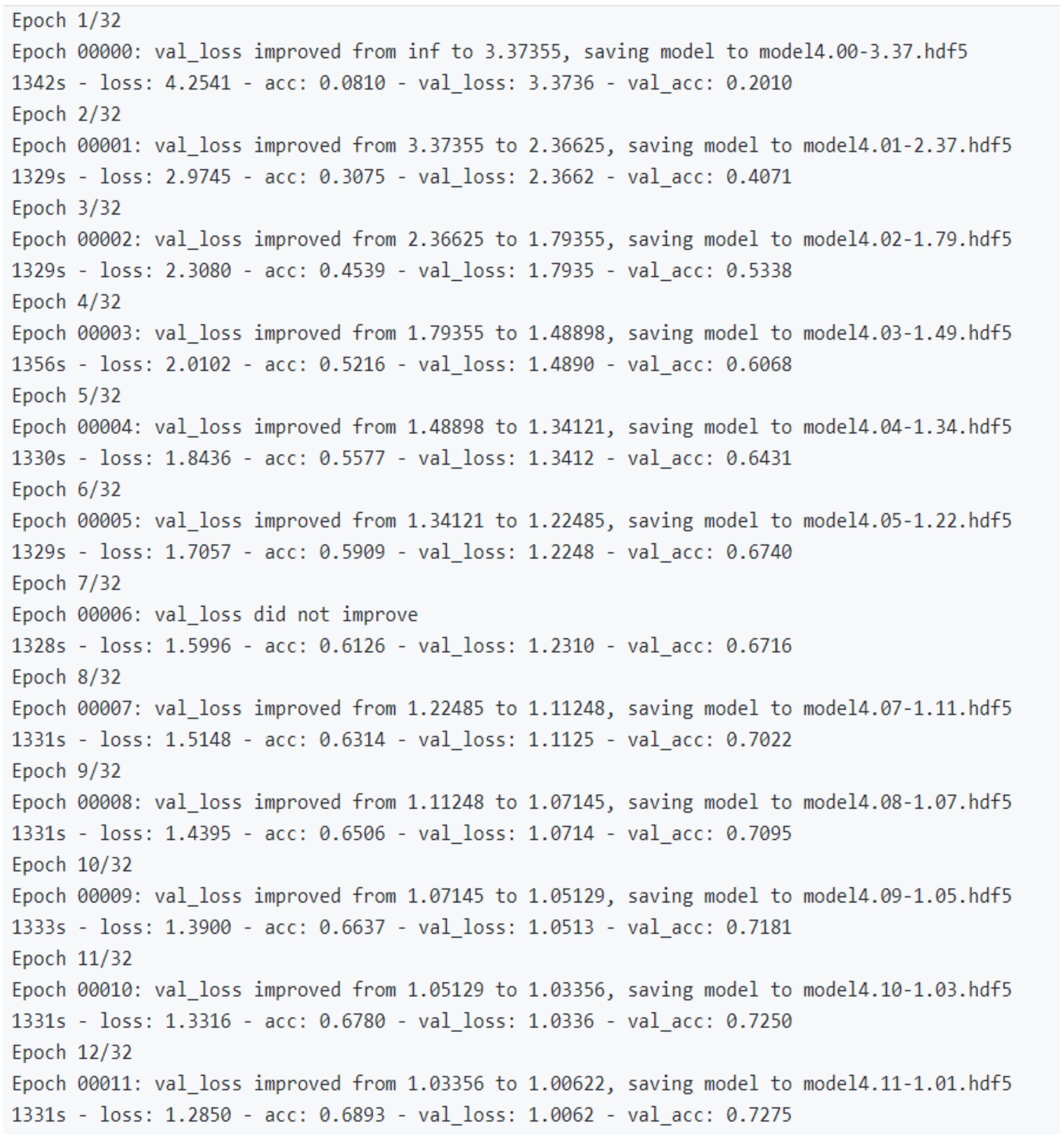
During the experimentation was performed multiple tests with differing structure of the network. Most accurate structure that was found was used to further experiment with meta-parameter settings. We have to mention that the first experiment on the dataset was done in a subset as it demanded lots of time resources for the training to complete. When we got close to the best values of different parameters such as dropout or learning rate, we used them to train the final model in the whole dataset. Coming next, the first training attempt of our whole dataset will be discussed.

First element that had to be selected was learning algorithm. Our initial experiments used more modern optimizers such as Adam, along with higher learning rates. The accuracy results were below 80%. The selected algorithm was Inception V3 with a quickly decreasing learning schedule. When we search through the multidimensional surface, sometimes going slower goes a long way.

The next parameter was the learning rate. Initially, in the first trained model we used a 0.001 value for learning rate with a 0.9 momentum. As for the Dropout value, in this first experiment, we chose the value of 0.4. The number of epochs chosen was 24. The batch size we initially chose was 128 but we very soon noticed that that the value wasn't proper so we decided the value of 64. Finally, as for the Dense we chose the value of 2048.

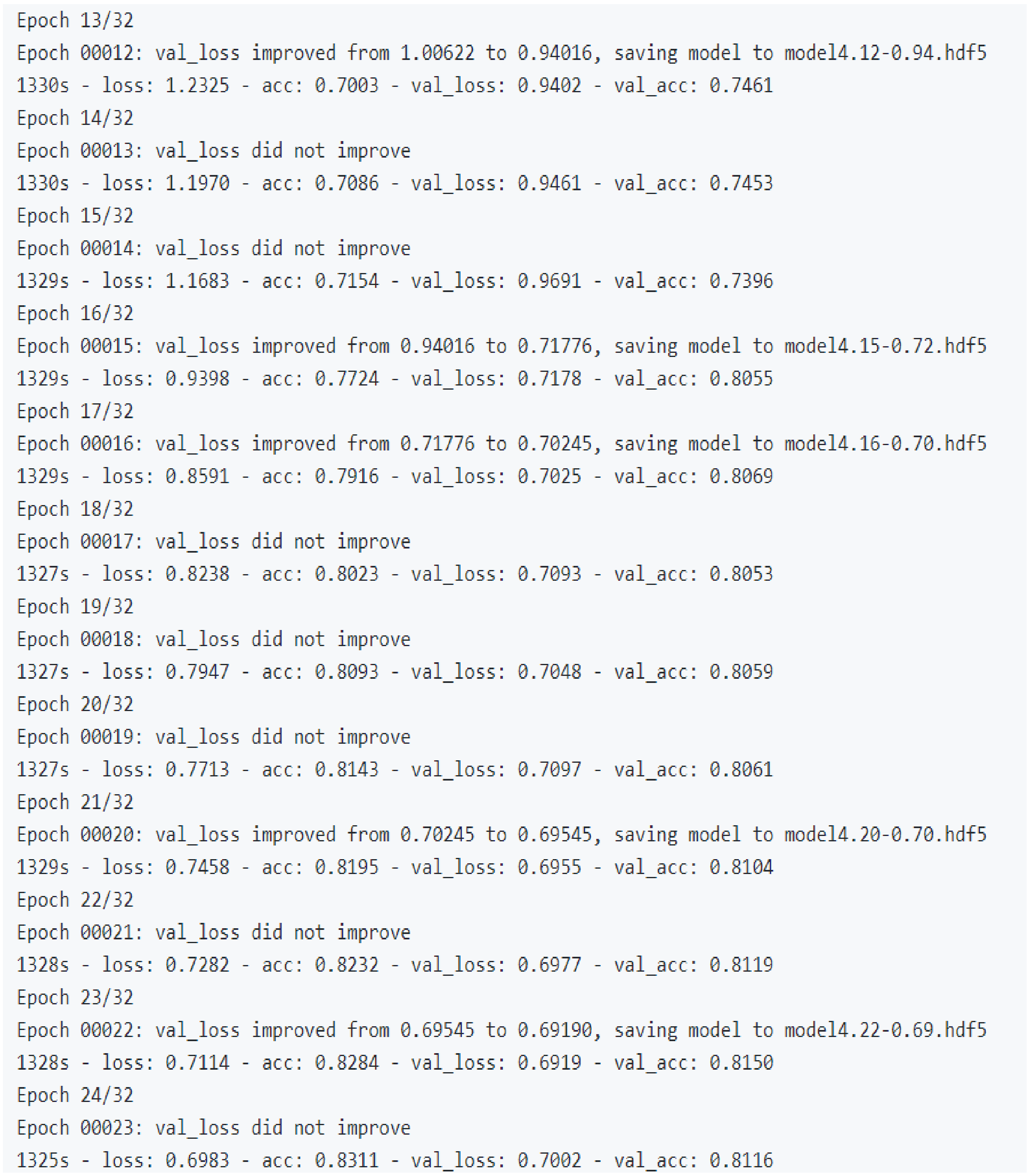
3.4.2 First Model Results

According to the values of these parameters the results are shown in Figure 3.1 and 3.2.



```
Epoch 1/32
Epoch 00000: val_loss improved from inf to 3.37355, saving model to model4.00-3.37.hdf5
1342s - loss: 4.2541 - acc: 0.0810 - val_loss: 3.3736 - val_acc: 0.2010
Epoch 2/32
Epoch 00001: val_loss improved from 3.37355 to 2.36625, saving model to model4.01-2.37.hdf5
1329s - loss: 2.9745 - acc: 0.3075 - val_loss: 2.3662 - val_acc: 0.4071
Epoch 3/32
Epoch 00002: val_loss improved from 2.36625 to 1.79355, saving model to model4.02-1.79.hdf5
1329s - loss: 2.3080 - acc: 0.4539 - val_loss: 1.7935 - val_acc: 0.5338
Epoch 4/32
Epoch 00003: val_loss improved from 1.79355 to 1.48898, saving model to model4.03-1.49.hdf5
1356s - loss: 2.0102 - acc: 0.5216 - val_loss: 1.4890 - val_acc: 0.6068
Epoch 5/32
Epoch 00004: val_loss improved from 1.48898 to 1.34121, saving model to model4.04-1.34.hdf5
1330s - loss: 1.8436 - acc: 0.5577 - val_loss: 1.3412 - val_acc: 0.6431
Epoch 6/32
Epoch 00005: val_loss improved from 1.34121 to 1.22485, saving model to model4.05-1.22.hdf5
1329s - loss: 1.7057 - acc: 0.5909 - val_loss: 1.2248 - val_acc: 0.6740
Epoch 7/32
Epoch 00006: val_loss did not improve
1328s - loss: 1.5996 - acc: 0.6126 - val_loss: 1.2310 - val_acc: 0.6716
Epoch 8/32
Epoch 00007: val_loss improved from 1.22485 to 1.11248, saving model to model4.07-1.11.hdf5
1331s - loss: 1.5148 - acc: 0.6314 - val_loss: 1.1125 - val_acc: 0.7022
Epoch 9/32
Epoch 00008: val_loss improved from 1.11248 to 1.07145, saving model to model4.08-1.07.hdf5
1331s - loss: 1.4395 - acc: 0.6506 - val_loss: 1.0714 - val_acc: 0.7095
Epoch 10/32
Epoch 00009: val_loss improved from 1.07145 to 1.05129, saving model to model4.09-1.05.hdf5
1333s - loss: 1.3900 - acc: 0.6637 - val_loss: 1.0513 - val_acc: 0.7181
Epoch 11/32
Epoch 00010: val_loss improved from 1.05129 to 1.03356, saving model to model4.10-1.03.hdf5
1331s - loss: 1.3316 - acc: 0.6780 - val_loss: 1.0336 - val_acc: 0.7250
Epoch 12/32
Epoch 00011: val_loss improved from 1.03356 to 1.00622, saving model to model4.11-1.01.hdf5
1331s - loss: 1.2850 - acc: 0.6893 - val_loss: 1.0062 - val_acc: 0.7275
```

Figure 3.1 Results of the first experiment



```
Epoch 13/32
Epoch 00012: val_loss improved from 1.00622 to 0.94016, saving model to model4.12-0.94.hdf5
1330s - loss: 1.2325 - acc: 0.7003 - val_loss: 0.9402 - val_acc: 0.7461
Epoch 14/32
Epoch 00013: val_loss did not improve
1330s - loss: 1.1970 - acc: 0.7086 - val_loss: 0.9461 - val_acc: 0.7453
Epoch 15/32
Epoch 00014: val_loss did not improve
1329s - loss: 1.1683 - acc: 0.7154 - val_loss: 0.9691 - val_acc: 0.7396
Epoch 16/32
Epoch 00015: val_loss improved from 0.94016 to 0.71776, saving model to model4.15-0.72.hdf5
1329s - loss: 0.9398 - acc: 0.7724 - val_loss: 0.7178 - val_acc: 0.8055
Epoch 17/32
Epoch 00016: val_loss improved from 0.71776 to 0.70245, saving model to model4.16-0.70.hdf5
1329s - loss: 0.8591 - acc: 0.7916 - val_loss: 0.7025 - val_acc: 0.8069
Epoch 18/32
Epoch 00017: val_loss did not improve
1327s - loss: 0.8238 - acc: 0.8023 - val_loss: 0.7093 - val_acc: 0.8053
Epoch 19/32
Epoch 00018: val_loss did not improve
1327s - loss: 0.7947 - acc: 0.8093 - val_loss: 0.7048 - val_acc: 0.8059
Epoch 20/32
Epoch 00019: val_loss did not improve
1327s - loss: 0.7713 - acc: 0.8143 - val_loss: 0.7097 - val_acc: 0.8061
Epoch 21/32
Epoch 00020: val_loss improved from 0.70245 to 0.69545, saving model to model4.20-0.70.hdf5
1329s - loss: 0.7458 - acc: 0.8195 - val_loss: 0.6955 - val_acc: 0.8104
Epoch 22/32
Epoch 00021: val_loss did not improve
1328s - loss: 0.7282 - acc: 0.8232 - val_loss: 0.6977 - val_acc: 0.8119
Epoch 23/32
Epoch 00022: val_loss improved from 0.69545 to 0.69190, saving model to model4.22-0.69.hdf5
1328s - loss: 0.7114 - acc: 0.8284 - val_loss: 0.6919 - val_acc: 0.8150
Epoch 24/32
Epoch 00023: val_loss did not improve
1325s - loss: 0.6983 - acc: 0.8311 - val_loss: 0.7002 - val_acc: 0.8116
```

Figure 3.2 Results of the first experiment

In this first attempt we achieved an accuracy of 0.8311 which we knew could surely be improved with different parameters. The val_accuracy and accuracy values are shown in Figure 3.3.

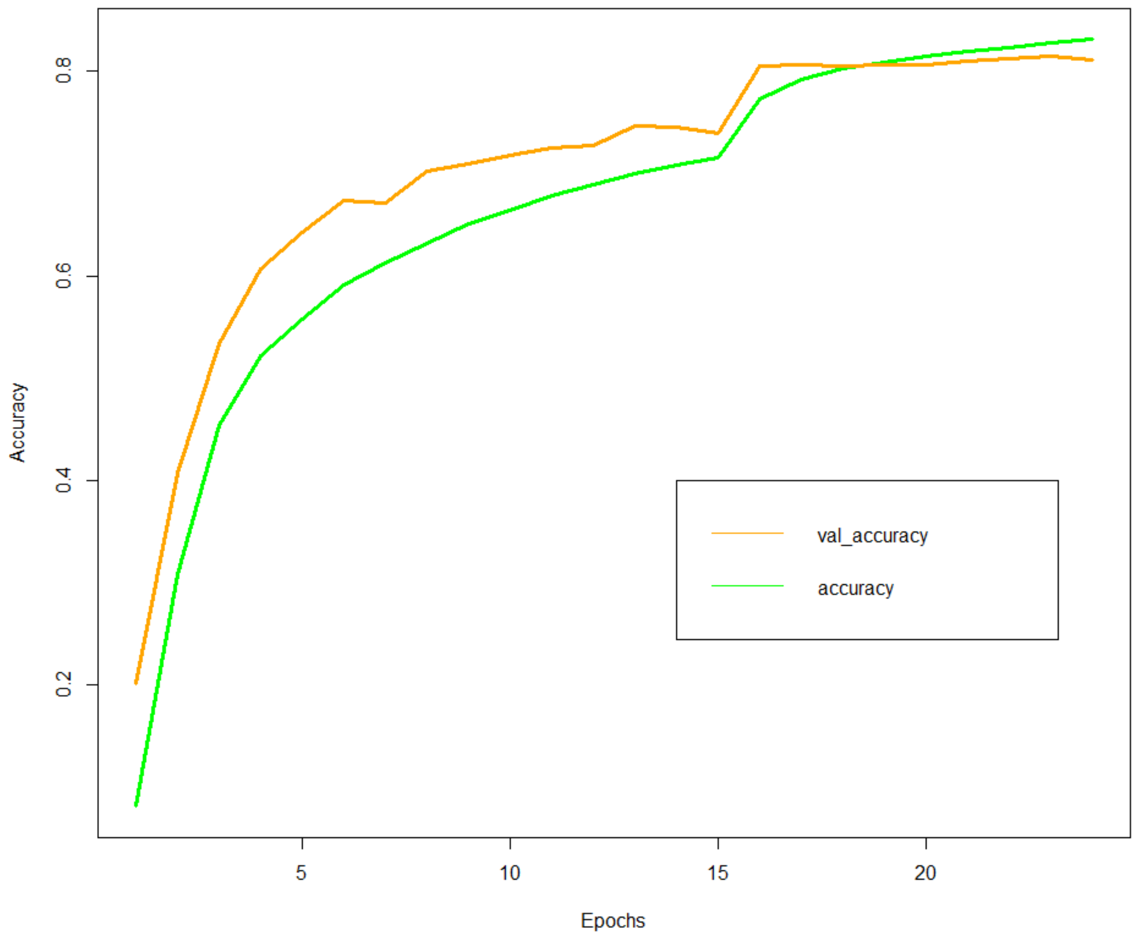
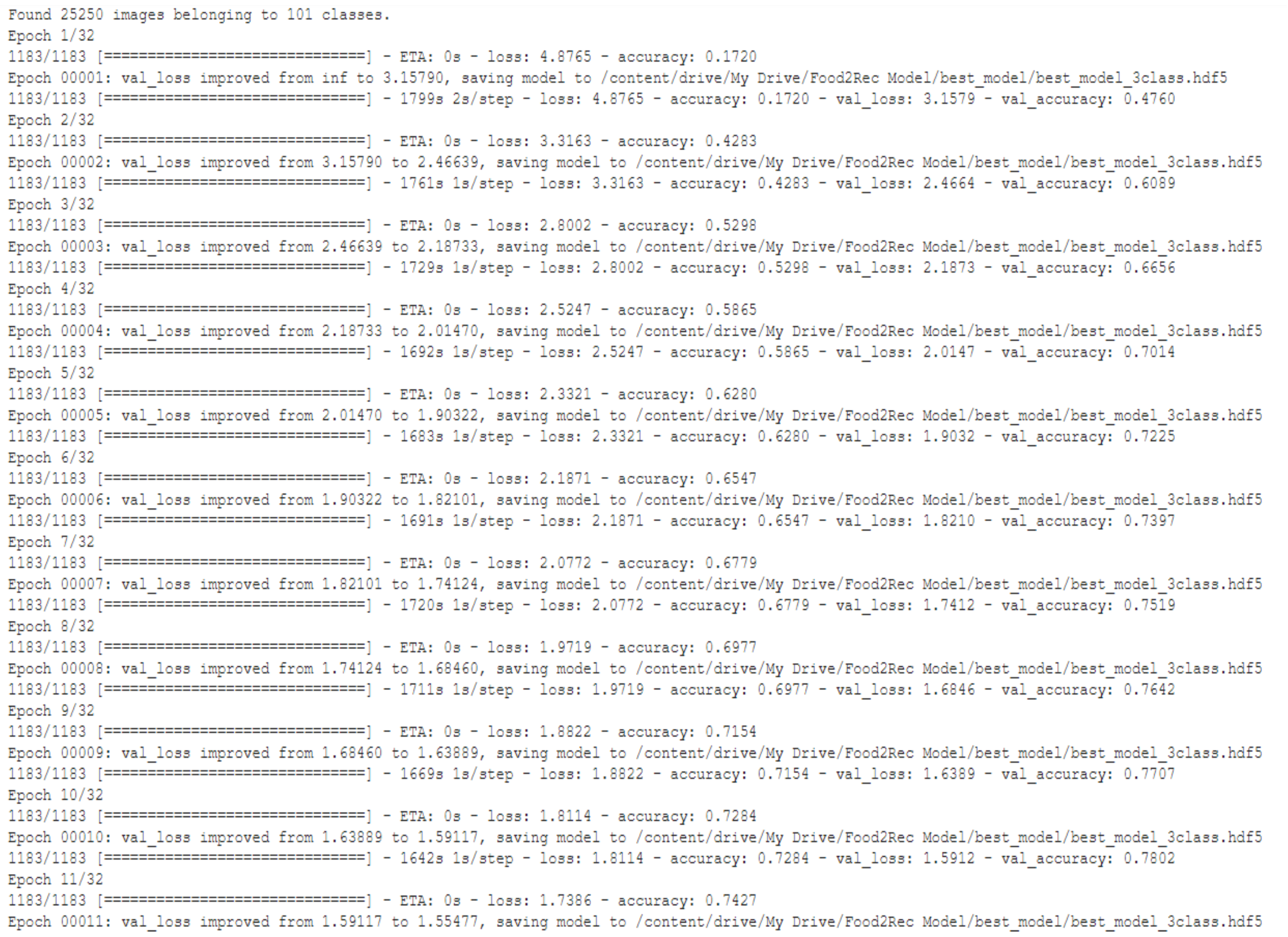


Figure 3.3 Accuracy in each Epoch

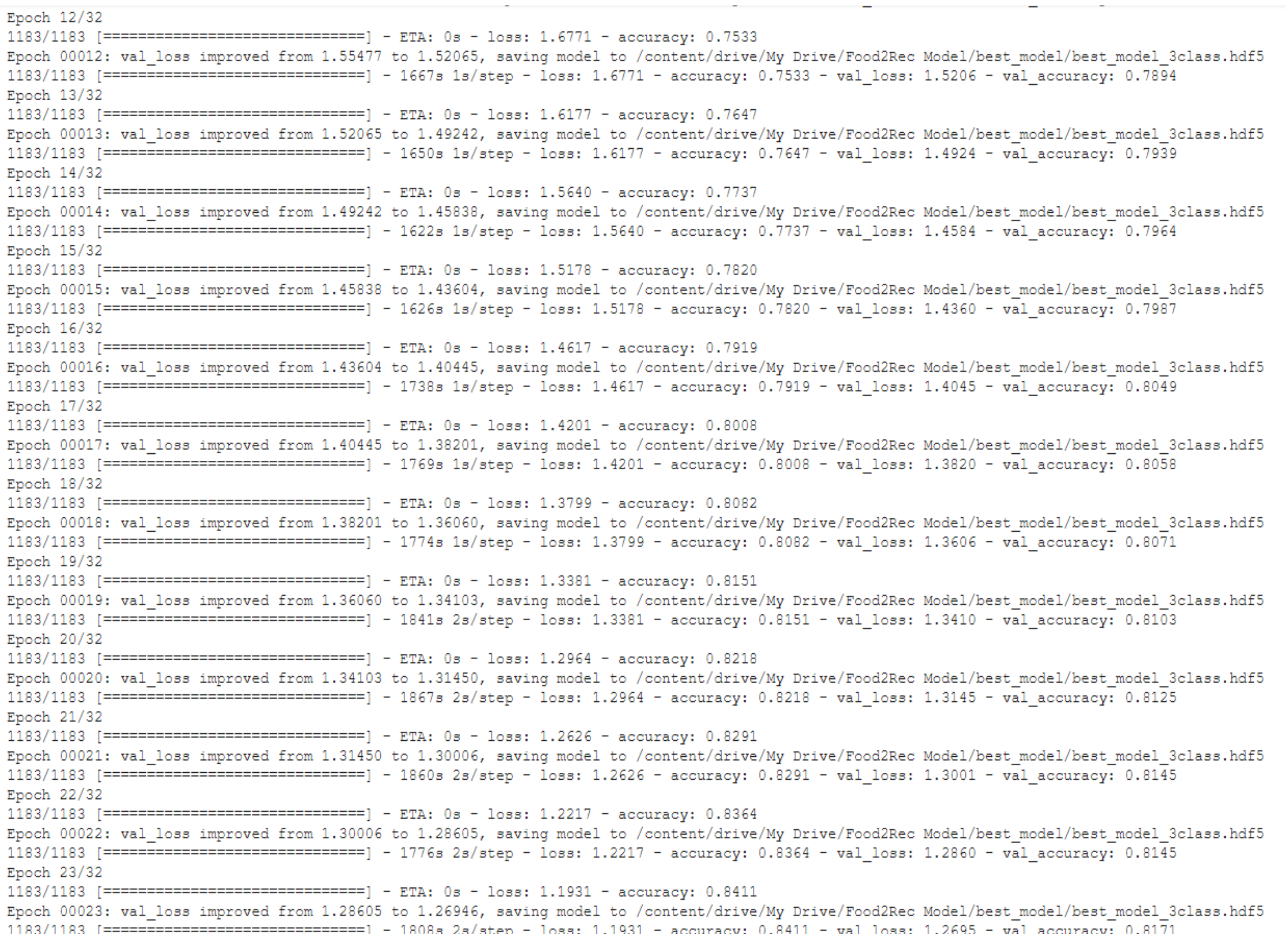
3.5 Final Model Results

As shown in a previous section, the final values in some parameters where slightly different. The batch size remained the same at 64. The value for Dropout selected was 0.5 and the learning rate value decreased to 0.01. As for the number of epochs, we increased it to 32 because we noticed that there was more space for our model to learn. Finally, we increased the number of Dense layers to 4096. The results are shown below.



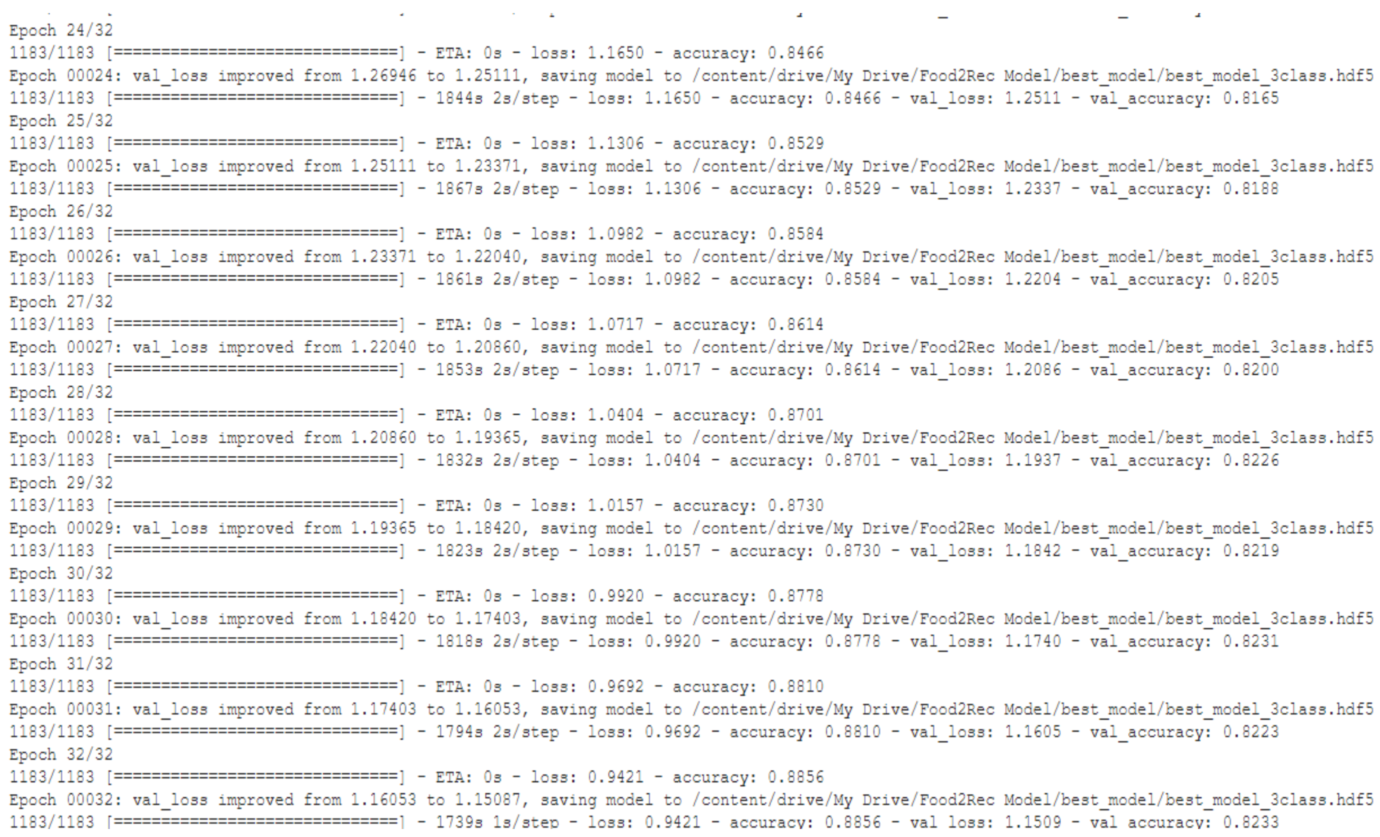
```
Found 25250 images belonging to 101 classes.
Epoch 1/32
1183/1183 [==============================] - ETA: 0s - loss: 4.8765 - accuracy: 0.1720
Epoch 00001: val_loss improved from inf to 3.15790, saving model to /content/drive/My Drive/Food2Rec Model/best_model/best_model_3class.hdf5
1183/1183 [==============================] - 1799s 2s/step - loss: 4.8765 - accuracy: 0.1720 - val_loss: 3.1579 - val_accuracy: 0.4760
Epoch 2/32
1183/1183 [==============================] - ETA: 0s - loss: 3.3163 - accuracy: 0.4283
Epoch 00002: val_loss improved from 3.15790 to 2.46639, saving model to /content/drive/My Drive/Food2Rec Model/best_model/best_model_3class.hdf5
1183/1183 [==============================] - 1761s 1s/step - loss: 3.3163 - accuracy: 0.4283 - val_loss: 2.4664 - val_accuracy: 0.6089
Epoch 3/32
1183/1183 [==============================] - ETA: 0s - loss: 2.8002 - accuracy: 0.5298
Epoch 00003: val_loss improved from 2.46639 to 2.18733, saving model to /content/drive/My Drive/Food2Rec Model/best_model/best_model_3class.hdf5
1183/1183 [==============================] - 1729s 1s/step - loss: 2.8002 - accuracy: 0.5298 - val_loss: 2.1873 - val_accuracy: 0.6656
Epoch 4/32
1183/1183 [==============================] - ETA: 0s - loss: 2.5247 - accuracy: 0.5865
Epoch 00004: val_loss improved from 2.18733 to 2.01470, saving model to /content/drive/My Drive/Food2Rec Model/best_model/best_model_3class.hdf5
1183/1183 [==============================] - 1692s 1s/step - loss: 2.5247 - accuracy: 0.5865 - val_loss: 2.0147 - val_accuracy: 0.7014
Epoch 5/32
1183/1183 [==============================] - ETA: 0s - loss: 2.3321 - accuracy: 0.6280
Epoch 00005: val_loss improved from 2.01470 to 1.90322, saving model to /content/drive/My Drive/Food2Rec Model/best_model/best_model_3class.hdf5
1183/1183 [==============================] - 1683s 1s/step - loss: 2.3321 - accuracy: 0.6280 - val_loss: 1.9032 - val_accuracy: 0.7225
Epoch 6/32
1183/1183 [==============================] - ETA: 0s - loss: 2.1871 - accuracy: 0.6547
Epoch 00006: val_loss improved from 1.90322 to 1.82101, saving model to /content/drive/My Drive/Food2Rec Model/best_model/best_model_3class.hdf5
1183/1183 [==============================] - 1691s 1s/step - loss: 2.1871 - accuracy: 0.6547 - val_loss: 1.8210 - val_accuracy: 0.7397
Epoch 7/32
1183/1183 [==============================] - ETA: 0s - loss: 2.0772 - accuracy: 0.6779
Epoch 00007: val_loss improved from 1.82101 to 1.74124, saving model to /content/drive/My Drive/Food2Rec Model/best_model/best_model_3class.hdf5
1183/1183 [==============================] - 1720s 1s/step - loss: 2.0772 - accuracy: 0.6779 - val_loss: 1.7412 - val_accuracy: 0.7519
Epoch 8/32
1183/1183 [==============================] - ETA: 0s - loss: 1.9719 - accuracy: 0.6977
Epoch 00008: val_loss improved from 1.74124 to 1.68460, saving model to /content/drive/My Drive/Food2Rec Model/best_model/best_model_3class.hdf5
1183/1183 [==============================] - 1711s 1s/step - loss: 1.9719 - accuracy: 0.6977 - val_loss: 1.6846 - val_accuracy: 0.7642
Epoch 9/32
1183/1183 [==============================] - ETA: 0s - loss: 1.8822 - accuracy: 0.7154
Epoch 00009: val_loss improved from 1.68460 to 1.63889, saving model to /content/drive/My Drive/Food2Rec Model/best_model/best_model_3class.hdf5
1183/1183 [==============================] - 1669s 1s/step - loss: 1.8822 - accuracy: 0.7154 - val_loss: 1.6389 - val_accuracy: 0.7707
Epoch 10/32
1183/1183 [==============================] - ETA: 0s - loss: 1.8114 - accuracy: 0.7284
Epoch 00010: val_loss improved from 1.63889 to 1.59117, saving model to /content/drive/My Drive/Food2Rec Model/best_model/best_model_3class.hdf5
1183/1183 [==============================] - 1642s 1s/step - loss: 1.8114 - accuracy: 0.7284 - val_loss: 1.5912 - val_accuracy: 0.7802
Epoch 11/32
1183/1183 [==============================] - ETA: 0s - loss: 1.7386 - accuracy: 0.7427
Epoch 00011: val_loss improved from 1.59117 to 1.55477, saving model to /content/drive/My Drive/Food2Rec Model/best_model/best_model_3class.hdf5
```

Figure 3.4 Best Model Results



```
Epoch 12/32
1183/1183 [==============================] - ETA: 0s - loss: 1.6771 - accuracy: 0.7533
Epoch 00012: val_loss improved from 1.55477 to 1.52065, saving model to /content/drive/My Drive/Food2Rec Model/best_model/best_model_3class.hdf5
1183/1183 [==============================] - 1667s 1s/step - loss: 1.6771 - accuracy: 0.7533 - val_loss: 1.5206 - val_accuracy: 0.7894
Epoch 13/32
1183/1183 [==============================] - ETA: 0s - loss: 1.6177 - accuracy: 0.7647
Epoch 00013: val_loss improved from 1.52065 to 1.49242, saving model to /content/drive/My Drive/Food2Rec Model/best_model/best_model_3class.hdf5
1183/1183 [==============================] - 1650s 1s/step - loss: 1.6177 - accuracy: 0.7647 - val_loss: 1.4924 - val_accuracy: 0.7939
Epoch 14/32
1183/1183 [==============================] - ETA: 0s - loss: 1.5640 - accuracy: 0.7737
Epoch 00014: val_loss improved from 1.49242 to 1.45838, saving model to /content/drive/My Drive/Food2Rec Model/best_model/best_model_3class.hdf5
1183/1183 [==============================] - 1622s 1s/step - loss: 1.5640 - accuracy: 0.7737 - val_loss: 1.4584 - val_accuracy: 0.7964
Epoch 15/32
1183/1183 [==============================] - ETA: 0s - loss: 1.5178 - accuracy: 0.7820
Epoch 00015: val_loss improved from 1.45838 to 1.43604, saving model to /content/drive/My Drive/Food2Rec Model/best_model/best_model_3class.hdf5
1183/1183 [==============================] - 1626s 1s/step - loss: 1.5178 - accuracy: 0.7820 - val_loss: 1.4360 - val_accuracy: 0.7987
Epoch 16/32
1183/1183 [==============================] - ETA: 0s - loss: 1.4617 - accuracy: 0.7919
Epoch 00016: val_loss improved from 1.43604 to 1.40445, saving model to /content/drive/My Drive/Food2Rec Model/best_model/best_model_3class.hdf5
1183/1183 [==============================] - 1738s 1s/step - loss: 1.4617 - accuracy: 0.7919 - val_loss: 1.4045 - val_accuracy: 0.8049
Epoch 17/32
1183/1183 [==============================] - ETA: 0s - loss: 1.4201 - accuracy: 0.8008
Epoch 00017: val_loss improved from 1.40445 to 1.38201, saving model to /content/drive/My Drive/Food2Rec Model/best_model/best_model_3class.hdf5
1183/1183 [==============================] - 1769s 1s/step - loss: 1.4201 - accuracy: 0.8008 - val_loss: 1.3820 - val_accuracy: 0.8058
Epoch 18/32
1183/1183 [==============================] - ETA: 0s - loss: 1.3799 - accuracy: 0.8082
Epoch 00018: val_loss improved from 1.38201 to 1.36060, saving model to /content/drive/My Drive/Food2Rec Model/best_model/best_model_3class.hdf5
1183/1183 [==============================] - 1774s 1s/step - loss: 1.3799 - accuracy: 0.8082 - val_loss: 1.3606 - val_accuracy: 0.8071
Epoch 19/32
1183/1183 [==============================] - ETA: 0s - loss: 1.3381 - accuracy: 0.8151
Epoch 00019: val_loss improved from 1.36060 to 1.34103, saving model to /content/drive/My Drive/Food2Rec Model/best_model/best_model_3class.hdf5
1183/1183 [==============================] - 1841s 2s/step - loss: 1.3381 - accuracy: 0.8151 - val_loss: 1.3410 - val_accuracy: 0.8103
Epoch 20/32
1183/1183 [==============================] - ETA: 0s - loss: 1.2964 - accuracy: 0.8218
Epoch 00020: val_loss improved from 1.34103 to 1.31450, saving model to /content/drive/My Drive/Food2Rec Model/best_model/best_model_3class.hdf5
1183/1183 [==============================] - 1867s 2s/step - loss: 1.2964 - accuracy: 0.8218 - val_loss: 1.3145 - val_accuracy: 0.8125
Epoch 21/32
1183/1183 [==============================] - ETA: 0s - loss: 1.2626 - accuracy: 0.8291
Epoch 00021: val_loss improved from 1.31450 to 1.30006, saving model to /content/drive/My Drive/Food2Rec Model/best_model/best_model_3class.hdf5
1183/1183 [==============================] - 1860s 2s/step - loss: 1.2626 - accuracy: 0.8291 - val_loss: 1.3001 - val_accuracy: 0.8145
Epoch 22/32
1183/1183 [==============================] - ETA: 0s - loss: 1.2217 - accuracy: 0.8364
Epoch 00022: val_loss improved from 1.30006 to 1.28605, saving model to /content/drive/My Drive/Food2Rec Model/best_model/best_model_3class.hdf5
1183/1183 [==============================] - 1776s 2s/step - loss: 1.2217 - accuracy: 0.8364 - val_loss: 1.2860 - val_accuracy: 0.8145
Epoch 23/32
1183/1183 [==============================] - ETA: 0s - loss: 1.1931 - accuracy: 0.8411
Epoch 00023: val_loss improved from 1.28605 to 1.26946, saving model to /content/drive/My Drive/Food2Rec Model/best_model/best_model_3class.hdf5
1183/1183 [==============================] - 1808s 2s/step - loss: 1.1931 - accuracy: 0.8411 - val_loss: 1.2695 - val_accuracy: 0.8171
```

Figure 3.5 Best Model Results



```
Epoch 24/32
1183/1183 [==============================] - ETA: 0s - loss: 1.1650 - accuracy: 0.8466
Epoch 00024: val_loss improved from 1.26946 to 1.25111, saving model to /content/drive/My Drive/Food2Rec Model/best_model/best_model_3class.hdf5
1183/1183 [==============================] - 1844s 2s/step - loss: 1.1650 - accuracy: 0.8466 - val_loss: 1.2511 - val_accuracy: 0.8165
Epoch 25/32
1183/1183 [==============================] - ETA: 0s - loss: 1.1306 - accuracy: 0.8529
Epoch 00025: val_loss improved from 1.25111 to 1.23371, saving model to /content/drive/My Drive/Food2Rec Model/best_model/best_model_3class.hdf5
1183/1183 [==============================] - 1867s 2s/step - loss: 1.1306 - accuracy: 0.8529 - val_loss: 1.2337 - val_accuracy: 0.8188
Epoch 26/32
1183/1183 [==============================] - ETA: 0s - loss: 1.0982 - accuracy: 0.8584
Epoch 00026: val_loss improved from 1.23371 to 1.22040, saving model to /content/drive/My Drive/Food2Rec Model/best_model/best_model_3class.hdf5
1183/1183 [==============================] - 1861s 2s/step - loss: 1.0982 - accuracy: 0.8584 - val_loss: 1.2204 - val_accuracy: 0.8205
Epoch 27/32
1183/1183 [==============================] - ETA: 0s - loss: 1.0717 - accuracy: 0.8614
Epoch 00027: val_loss improved from 1.22040 to 1.20860, saving model to /content/drive/My Drive/Food2Rec Model/best_model/best_model_3class.hdf5
1183/1183 [==============================] - 1853s 2s/step - loss: 1.0717 - accuracy: 0.8614 - val_loss: 1.2086 - val_accuracy: 0.8200
Epoch 28/32
1183/1183 [==============================] - ETA: 0s - loss: 1.0404 - accuracy: 0.8701
Epoch 00028: val_loss improved from 1.20860 to 1.19365, saving model to /content/drive/My Drive/Food2Rec Model/best_model/best_model_3class.hdf5
1183/1183 [==============================] - 1832s 2s/step - loss: 1.0404 - accuracy: 0.8701 - val_loss: 1.1937 - val_accuracy: 0.8226
Epoch 29/32
1183/1183 [==============================] - ETA: 0s - loss: 1.0157 - accuracy: 0.8730
Epoch 00029: val_loss improved from 1.19365 to 1.18420, saving model to /content/drive/My Drive/Food2Rec Model/best_model/best_model_3class.hdf5
1183/1183 [==============================] - 1823s 2s/step - loss: 1.0157 - accuracy: 0.8730 - val_loss: 1.1842 - val_accuracy: 0.8219
Epoch 30/32
1183/1183 [==============================] - ETA: 0s - loss: 0.9920 - accuracy: 0.8778
Epoch 00030: val_loss improved from 1.18420 to 1.17403, saving model to /content/drive/My Drive/Food2Rec Model/best_model/best_model_3class.hdf5
1183/1183 [==============================] - 1818s 2s/step - loss: 0.9920 - accuracy: 0.8778 - val_loss: 1.1740 - val_accuracy: 0.8231
Epoch 31/32
1183/1183 [==============================] - ETA: 0s - loss: 0.9692 - accuracy: 0.8810
Epoch 00031: val_loss improved from 1.17403 to 1.16053, saving model to /content/drive/My Drive/Food2Rec Model/best_model/best_model_3class.hdf5
1183/1183 [==============================] - 1794s 2s/step - loss: 0.9692 - accuracy: 0.8810 - val_loss: 1.1605 - val_accuracy: 0.8223
Epoch 32/32
1183/1183 [==============================] - ETA: 0s - loss: 0.9421 - accuracy: 0.8856
Epoch 00032: val_loss improved from 1.16053 to 1.15087, saving model to /content/drive/My Drive/Food2Rec Model/best_model/best_model_3class.hdf5
1183/1183 [==============================] - 1739s 1s/step - loss: 0.9421 - accuracy: 0.8856 - val_loss: 1.1509 - val_accuracy: 0.8233
```

Figure 3.6 Best Model Results

In this final attempt we achieved an accuracy of 0.8856, a value we knew it was very good for such a large dataset. The val_accuracy and accuracy values are shown in Figure 3.7.

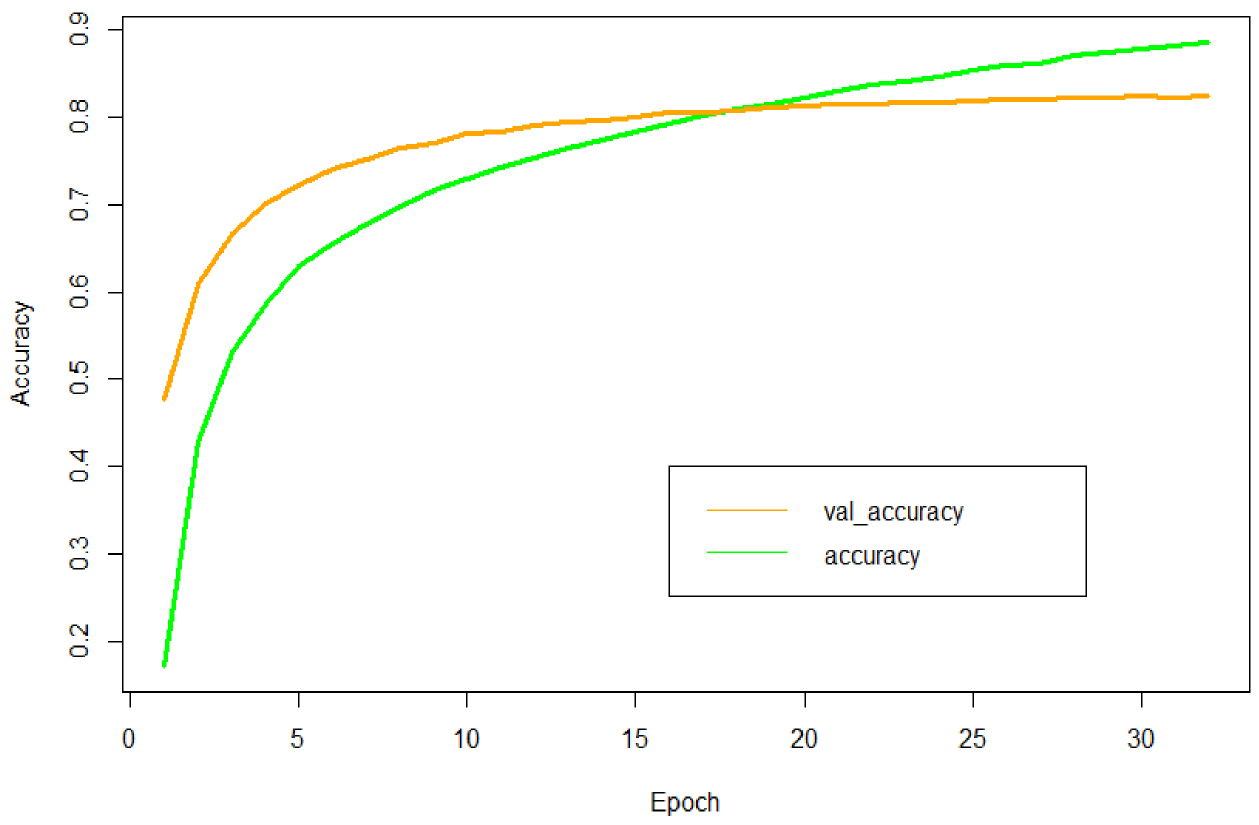


Figure 3.7 Accuracy in each Epoch

3.6 Conclusion

The best model was the one with 0.88 accuracy which was better than the first trained model with 0.83 accuracy. The best number of epochs is 32, since the validation accuracy falls in more epochs indicating that our model is starting to fit on noise and is beginning to over fit, losing the ability to predict new data. Training the model wasn't an easy work. Lots of time had to be spent in order for completion. This fact was the reason we didn't experiment a lot more with the different parameters and values as the process was heavy. The final and best model train took almost 14 hours for all 32 epochs to complete. We were very satisfied with the accuracy achieved for our model as some manual tests for the predictions of the model were made and led to the conclusion that the model was well trained. Many different parameters were used in order to train different models. MobileNetV2 was used as the main algorithm instead of InceptionV3 but the results were very similar, therefore InceptionV3 was chosen. After this whole process, the next part was the implementation in the application.

4. Application Analysis

4.1 Requirements

This section will analyze the system requirements. Application requirements can be divided into functional and non-functional requirements. Functional requirements define the capabilities and functions that the system should be able to perform successfully. Non-functional requirements are defined as criteria and properties that can be used to evaluate system performance.

4.1.1 Functional Requirements

Below are the functional requirements of the application.

1. Users will be able to create an account in the application.
2. Users will be able to sign in using either their Gmail or Facebook account.
3. Users will be able to log out of their account, whoever it is.
4. Users will be able to verify their email after creating an account.
5. Users will be able to retrieve the application code via their email.
6. Users will be able to state a name that represents them in the application and will be able to change it through the application after logging in.
7. Users will be able to access the device camera and gallery in order to choose a picture.
8. Users will be able to upload the picture in order for it to be analyzed and be recognized by the Machine Learning model trained.
9. Users will be able to choose one of the results of the recognized picture.
10. Users will be able to navigate to a screen that contain the results and all the different recipes.
11. Users will add every recipe of the successfully recognized dish in their history in order to access any time.
12. Users will be able to delete the history.
13. Users will be able to search for desired recipes in the application's database and access them with all the necessary information.
14. Users will be able to rate any recipe.
15. Users will be able to add a recipe to their favorites.
16. Users will be able to download any recipe to their mobile phone as a PDF file.

4.1.2 Non-Functional Requirements

Listed below are the non-functional requirements based on their importance:

- Usability: is the ease with which a user can learn to operate, prepare inputs and interpret system output.
- Modification / Scalability: The ability of the system to change easily to meet new requirements.
- Portability: The ease with which a system or component can be transferred from one environment to another.

- Flexibility: The ability of the system to easily exchange information with the user.
- Reliability: is the ability of a system to perform its required functions according to the conditions for a specific period of time.
- Security: login, passwords.
- Performance: refers to the operating speed of a system.
- Total cost: The total cost of the project in terms of price and time.

The non-functional requirements of our application are:

- The graphical interface of the application should be designed to provide a pleasant feeling to the user and will be easy to use.
- Alerts should be clear and restricted so as not to repel the user.
- The application should have no errors in the code and the user should be informed of any errors and the reason behind it.
- Database operations and interactions should be performed in the best possible way so as not to burden the performance and speed of the application.

4.2 Design

All the basic functions of the application and the data that have been collected or will be stored by the users, happen with the help of Firebase. Recipe information is stored in the Fire Store while new users are stored as well. Images/Videos used in the application (user's avatar, post, comments)are stored in Storage. A basic structure of the application is shown in Figure 3.1.

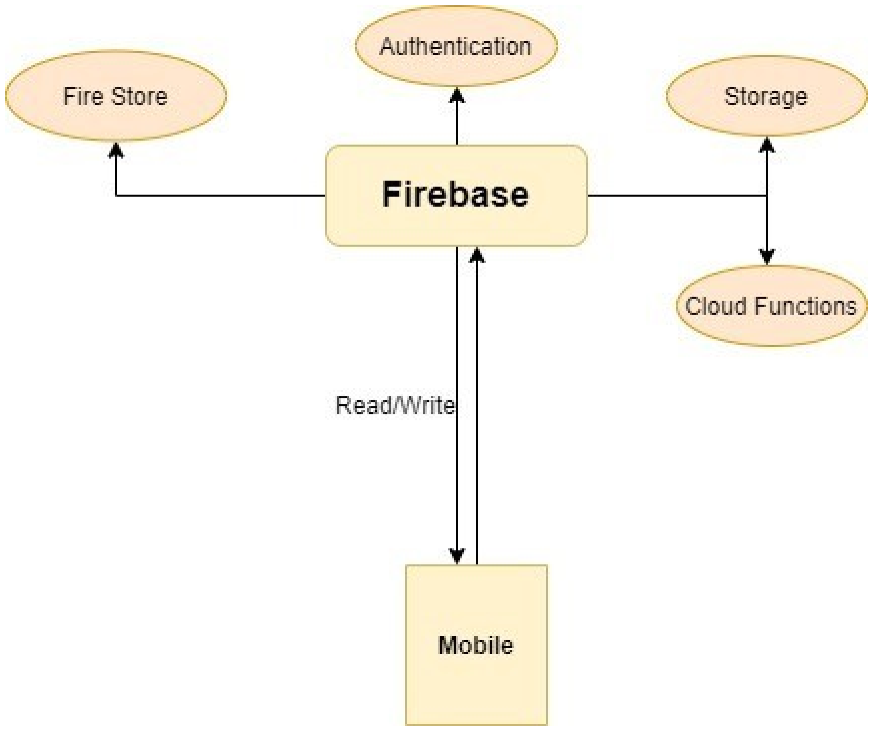


Figure 3.1 Basic Design

4.3 Database Structure

The database consists of collections. Its structure is presented below.

- Collection Users. Here the users information is stored.

  Users(Collection)

  Uid(Document)

  {

  activity: array,

  androidNotificationToken: String,

  displayName: String

  email: String,

  favouriteRecipes: array,

  followers: array,

  following: array,

```
        id: String,

        photoUrl: String,

        provider: String

        }
```

Each field and its description is shown in Figure 3.2.

activity: array	A list of all notification id's in order to display to the user
androidNotificationToken: String	The token needed for displaying notifications
displayName: String	User's in-app name
email: String	User's email
favouriteRecipes: array	List of the favorite recipes a user has selected
followers: array	List of users followed by
following: array	List of users following
id: String	Unique identification code
bio: String	User's description in profile
birthDate: Date	User's birth date
photoUrl: String	Photo url of the user's profile picture
provider: String	Refers to the registration provider(Google, Facebook, Email/Password)

Figure 3.2 users collection

- Collection recipes. The recipes information is stored here.

```
cooeat_recipes(Collection)
        chef(Document)
                category(collection)
                        recipe_title(Document)
                        {
                        cooking_method: array,
                        cooking_time: array,
                        ingredients: array
                        image_url: String,
                        rating: number,
                        recipe_title: String,
                        recipe_url: String,
                        users_rating: array,
                        }
```

cooking_method: array	The method described for the recipe
cooking_time: array	The cooking time fro the recipe
ingredients: array	A list of ingredients of the recipe
image_url: String	The image url for the recipe
rating: number	The total rating of the users for the recipe
recipe_title: String	The title of the recipe

recipe_url: String	The url that leads to the original site the recipe was taken from
users_rating: array	A list that contains all user id's that rated the recipe

Figure 3.3 recipes collection

5. Application Development

This section will present the application development process, code snippets and the techniques used.

5.1 Create Firebase Project

After installing Android Studio, the Firebase Project called FoodApp was created. At https://console.firebase.google.com/ and after logging in to your Google Account, we add a new Project as shown in Figure 4.1.

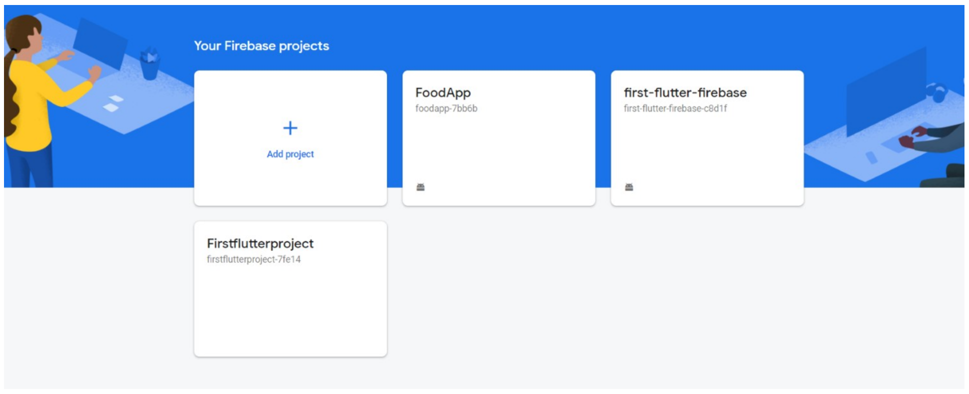


Figure 4.1 Creating Project

Then, after selecting the account to save the Project and having the Google Analytics option selected, which will provide charts for using the database, the Project is created after some time.To add an application to Project, select the Android icon as shown in Figure 4.2.

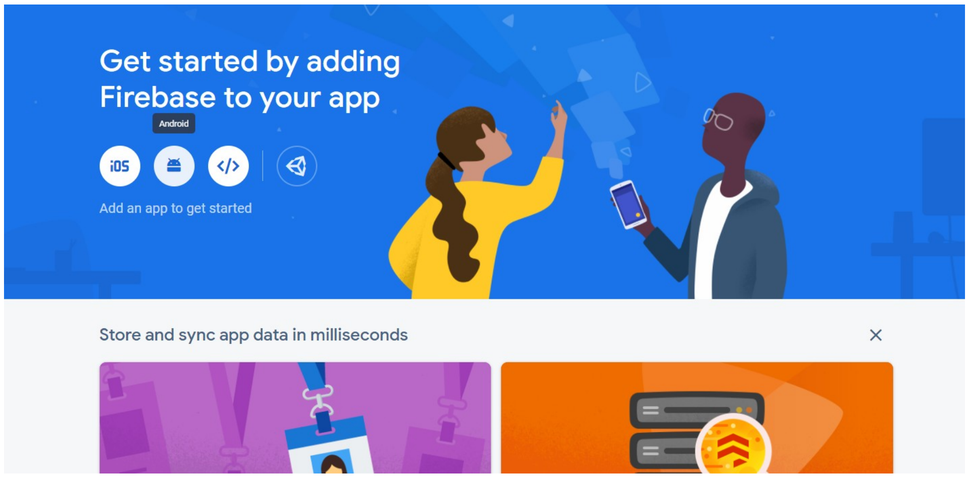

Get started by adding Firebase to your app
Android
iOS
</>
Add an app to get started
Store and sync app data in milliseconds

Figure 4.2 Add application

Next, select the application and package name as shown in Figure 4.3.

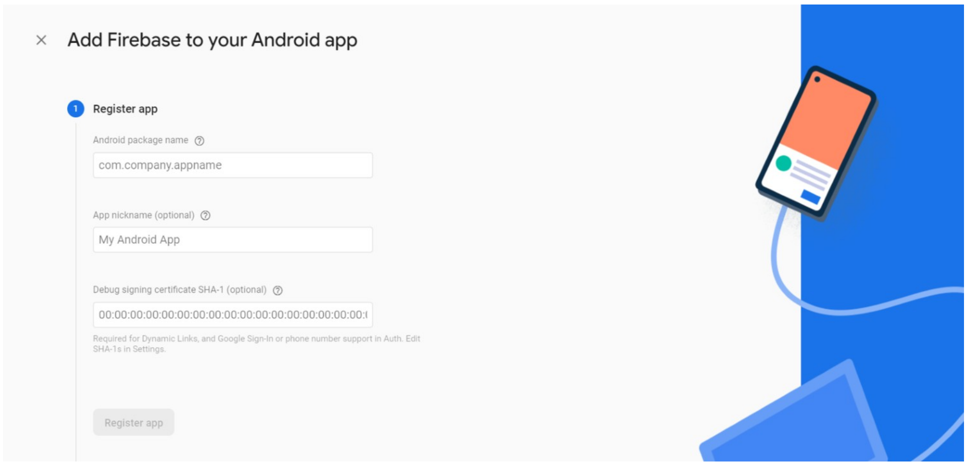
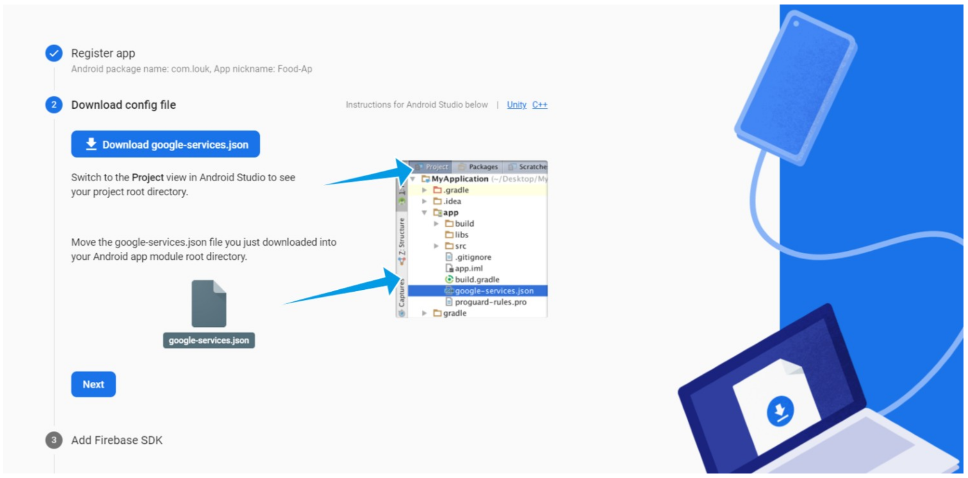

Add Firebase to your Android app
1 Register app
Android package name
com.company.appname
App nickname (optional)
My Android App
Debug signing certificate SHA-1 (optional)
00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00
Required for Dynamic Links, and Google Sign-In or phone number support in Auth. Edit SHA-1s in Settings.
Register app

Figure 4.3 Selecting an application name

After this step, a google-services.json file is generated which has all the necessary information to connect Android Studio to Firebase and should be placed in the path suggested as shown in Figure 4.4.


Register app

Android package name: com.louk, App nickname: Food-Ap

Download config file

Instructions for Android Studio below | Unity C++

Download google-services.json

Switch to the **Project** view in Android Studio to see your project root directory.

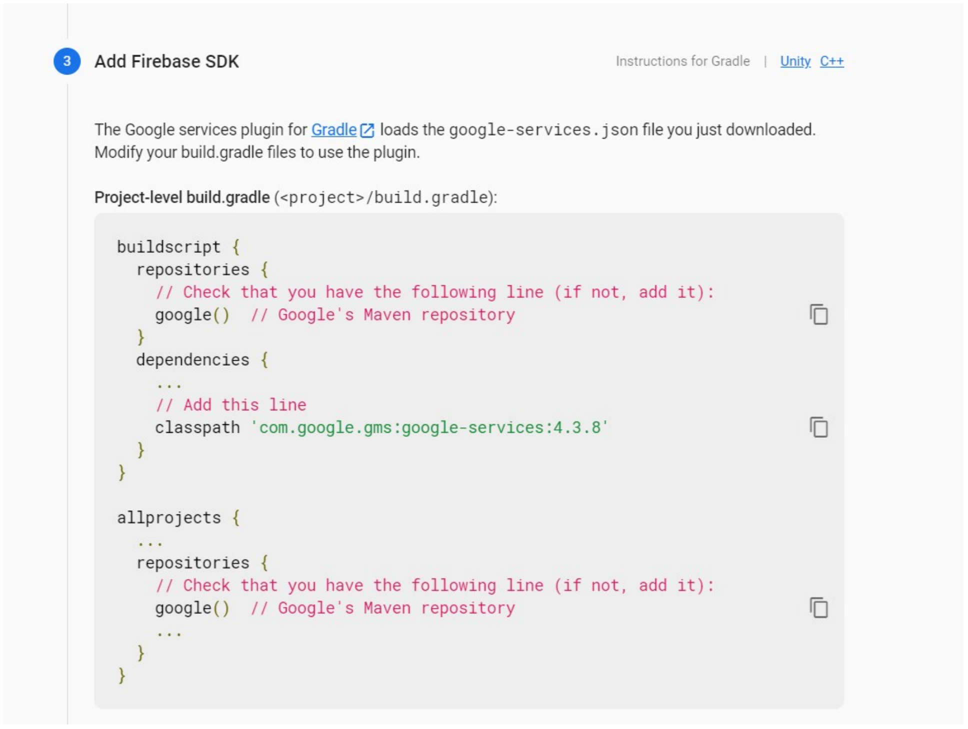
Move the google-services.json file you just downloaded into your Android app module root directory.

Next

Add Firebase SDK

Figure 4.4 The google-services.json file

Finally, the code series should be added to the Android Studio Project in the files suggested in Figures 4.5, 4.6.


Add Firebase SDK

Instructions for Gradle | Unity C++

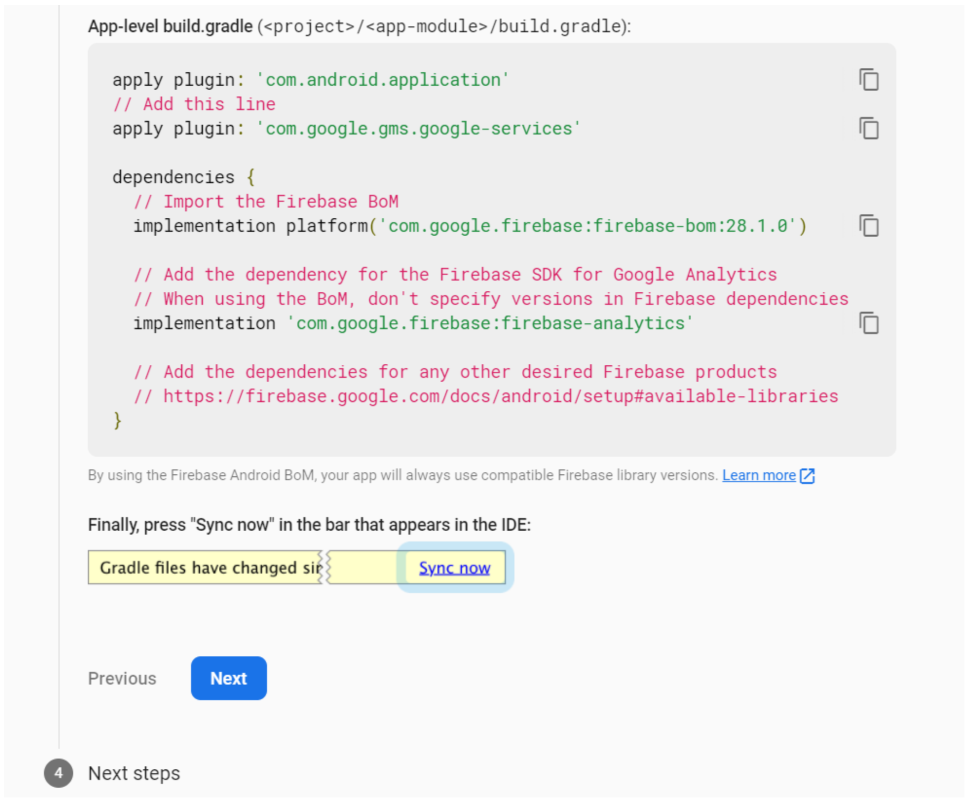
The Google services plugin for Gradle loads the `google-services.json` file you just downloaded. Modify your build.gradle files to use the plugin.

Project-level build.gradle (`<project>/build.gradle`):

```
buildscript {
  repositories {
    // Check that you have the following line (if not, add it):
    google()  // Google's Maven repository
  }
  dependencies {
    ...
    // Add this line
    classpath 'com.google.gms:google-services:4.3.8'
  }
}

allprojects {
  ...
  repositories {
    // Check that you have the following line (if not, add it):
    google()  // Google's Maven repository
    ...
  }
}
```

Figure 4.5


App-level build.gradle (<project>/<app-module>/build.gradle):

```
apply plugin: 'com.android.application'
// Add this line
apply plugin: 'com.google.gms.google-services'

dependencies {
  // Import the Firebase BoM
  implementation platform('com.google.firebase:firebase-bom:28.1.0')

  // Add the dependency for the Firebase SDK for Google Analytics
  // When using the BoM, don't specify versions in Firebase dependencies
  implementation 'com.google.firebase:firebase-analytics'

  // Add the dependencies for any other desired Firebase products
  // https://firebase.google.com/docs/android/setup#available-libraries
}
```

By using the Firebase Android BoM, your app will always use compatible Firebase library versions. Learn more

Finally, press "Sync now" in the bar that appears in the IDE:

Gradle files have changed sir Sync now

Figure 4.6

After adding the code, the Firebase Project and that of Android Studio are now connected.

5.2 Developing Process

As mentioned in the previous chapter, in order to fulfill the requirements, the application was splitted in different components. These are the authentication which includes the log in and sign up screen, the home screen that includes the socializing part of the application, the recipes search as well as any in app and system notifications.

5.2.1 Authentication Component

The general idea was to build a screen that includes different ways of authentication in order to log in in our application as shown in Figure 4.7.

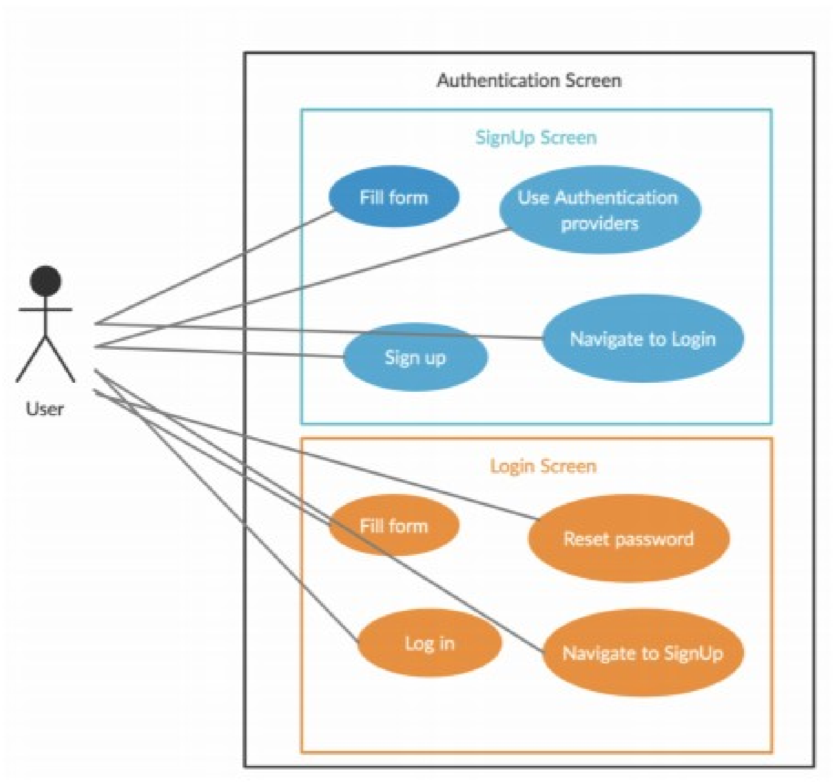


Figure 4.7

Three different ways that a user can sign up or sign in was selected. These are the following:

- Sign Up/ Sign In with Google.

- Sign Up/Sign In with Facebook.

- Sign Up/Sign In with Email/Password.

5.2.1.1 Sign Up/ Sign In with Google

To achieve google sign up with Firebase, a pop-up window displays after users select this sign-up option. After choosing the preferred email, a different page is presented and users are asked to fill a display name and pick a profile picture (if they want to). After the completion of the form, some actions are triggered.

The first action refers to the upload of user's photo for his profile picture in Firebase Storage with the _uploadGoogleImageToFirebaseStorage function. The content of this function is shown below:

```
var firebaseStorageRef = FirebaseStorage.instance
   .ref()
   .child('Google/${widget.gMail}/ProfilePic.png');
var uploadTask = firebaseStorageRef.putFile(utils.imageFileToFireStoreGoogle);
await uploadTask.then((res) async {
  url = await res.ref.getDownloadURL();
});
```

When this function completes, the url that refers to the path of the photo is saved in order to be used afterwards.

After the completion of the previous function the user's data are uploaded in Firestore. To achieve that, the _addGoogleUser function is triggered. The different fields such as id, email, displayName, as well as the photoUrl that is included in the url variable we saved with the _uploadGoogleImageToFirebaseStorage function, and others, are added to the users collection, inside a document that belongs to the specific user. The code sample is shown below:

```
FirebaseFirestore.instance.collection('users')
 .doc("${utils.auth.currentUser.uid}")
 .set({
 "id": utils.auth.currentUser.uid,
 'email': widget.gMail,
 'displayName': _googleDisplayName,
 'photoUrl': url,
 'provider': "google",
 'followers':[],
 'following':[],
 "favouriteRecipes":[],
 "activity":[],
 "androidNotificationToken":tokenRef

})
```

After this step, the user is navigated to the home screen.

In order for the user to sign in with Google they have to choose an existing email that they have already filled the form and completed the registration. The googleSignIn method is triggered at this occasion. The content of the method is shown after:

```
final GoogleSignInAccount googleSignInAccount =
await utils.googleSignIn.signIn();
final GoogleSignInAuthentication googleSignInAuthentication =
await googleSignInAccount.authentication;

final AuthCredential _credential = GoogleAuthProvider.credential(
  accessToken: googleSignInAuthentication.accessToken,
  idToken: googleSignInAuthentication.idToken,
);
```

This function also includes checks for whether a user has already an account for the specific email by a different method, meaning either Facebook or Email/Password. To achieve that, a call in Firestore is performed. In case the user owns an existing account for that email, a message appears on the screen to inform him. The code sample of this process is described below:

```
final search = await FirebaseFirestore.instance
    .collection('users')
    .where('email', isEqualTo: '$_gMail')
    .get();
var provider;
search.docs.forEach((res) {
  provider = res.data()["provider"];
});

if (provider == "email" || provider == "facebook") {
  toastMessages("Email already in use for another account.");
  utils.googleSignIn.signOut();
  _forkProgressIndicator().hide();
  this._loginForkController.dispose();
} else {
  if (search.docs.isEmpty) {
    return _goToGoogleScreen(
        context, _gMail, _gName, _gPhotoUrl, _credential);
  } else {
    try {

      Navigator.push(context,PageTransition(type:PageTransitionType.fade,alignment:Alignment.bottomCenter,
child:splash.SplashScreenMain()));
      await FirebaseAuth.instance
          .signInWithCredential(_credential);

    } catch (e) {

      toastMessages("Something went wrong");

    }
  }
}
```

5.2.1.2 Sign Up/ Sign In with Facebook

A very similar process to Google Sign Up/Sign In is followed for this case also. The main difference in order to accomplish Facebook sign up in Flutter is that developers have to create an account in https://developers.facebook..com and link the specific project to this account. After we created the account and linked the project we were able to add the Facebook sign up process in the application.

To achieve Facebook sign up with Firebase, a pop-up window displays after users select this sign-up option. After choosing their Facebook linked email, a different page is presented and users are asked to fill a display name and pick a profile picture (if they want to). After the completion of the form, the same actions as Google sign up are triggered.

The first action refers to the upload of user's photo for his profile picture in Firebase Storage with the _uploadFacebookImageToFirebaseStorage function. The content of this function is shown below:

```
var firebaseStorageRef = FirebaseStorage.instance
    .ref()
    .child('Facebook/${widget.fbProfile["email"]}/ProfilePic.png');
var uploadTask = firebaseStorageRef.putFile(utils.imageFileToFireStoreFB);
await uploadTask.then((res) async {
  url = await res.ref.getDownloadURL();
  print(url);
});
```

After the completion of the previous function the user's data are uploaded in Firestore. To achieve that, the _addFacebookUser function is triggered. The different fields such as id, email, displayName, as well as the photoUrl that is included in the url variable we saved with the _uploadFacebookImageToFirebaseStorage function, and others, are added to the users collection, inside a document that belongs to the specific user. The code sample is shown below:

```
utils.users
    .doc("${utils.auth.currentUser.uid}")
    .set({
  "id": utils.auth.currentUser.uid,
  'email': widget.fbProfile["email"], // John Doe
  'displayName': _fbDisplayName, // Stokes and Sons
  'photoUrl': url, //widget.fbProfile["picture"]["data"]["url"] // 42
  'provider': "facebook",
  'followers':[],
  'following':[],
  "favouriteRecipes":[],
  "activity":[],
  "androidNotificationToken":tokenRef
})
```

After this step, the user is navigated to the home screen.

In order for the user to sign in with Facebook they have to choose an existing email that they have already filled the form and completed the registration. The facebookSignIn method is triggered at this occasion. The content of the method is shown after:

```
final FacebookLoginResult result = await utils.fbLogin.logIn(["email"]);
switch (result.status) {
  case FacebookLoginStatus.loggedIn:
    final token = result.accessToken.token;
    final response=await http.get(Uri.parse('https://graph.facebook.com/v2.12/me?
fields=name,picture.width(800).height(800),first_name,last_name,email&access_token=${token}'));
    final profile = jsonDecode(response.body);
    print(profile);
    final search = await FirebaseFirestore.instance
        .collection('users')
        .where('email', isEqualTo: '${profile["email"]}')
        .get();
    var provider;
    search.docs.forEach((res) {
      provider = res.data()["provider"];
    });
    if (provider == "email" || provider == "google") {
      toastMessages("Email already in use for another account.");
      _forkProgressIndicator().hide();
      this._loginForkController.dispose();
    } else {
      if (search.docs.isEmpty) {
        _goToFbScreen(context, profile, token);
      } else {
        try {

          final AuthCredential credentials =
          FacebookAuthProvider.credential(token);
          _forkProgressIndicator().hide();
          _loginForkController.dispose();
          Navigator.push(context, PageTransition(type: PageTransitionType.fade,alignment:Alignment.bottomCenter, child:
splash.SplashScreenMain()));
          await FirebaseAuth.instance
              .signInWithCredential(credentials);


        } catch (e) {
          print(e);
          toastMessages(e.code);
        }
      }
    }
    break;
```

5.2.1.3 Sign Up/ Sign In with Email/Password

The last sign-up method is sign up with Email and Password. If a user selects this option, a different screen appears and the user has to fill a form providing his personal email, a password and the preferred display name. After the completion of the form the user is created in Firebase and data uploaded in Firestore similar to the other sign-up options. After this step, a verification email is sent and users must follow the provided link in that email. In addition, in order to send an email verification, user must be signed in. This was a problem for the application because we check for the state of the user, if he is signed in or not, in order to either push him in the home screen or the log in screen. To get over such a problem we sign in the user, send the verification email, upload user's data in Firestore and then sign out. Also checks if the password is too weak or the email already exists or a field is empty, are provided and messages appear in any case. This process is shown below:

```
await FirebaseAuth.instance
     .createUserWithEmailAndPassword(
     email: _signUpEmailController.text,
     password: _signUpPassController.text)
     .then((value) async{
   User user = FirebaseAuth.instance.currentUser;


   await _uploadEmailImageToFirebaseStorage();
   if (!user.emailVerified) {
     utils.auth.currentUser.sendEmailVerification();
     await _addEmailUser();
     utils.auth.signOut();


     Navigator.of(context).pop();
     FocusScope.of(context).requestFocus(new FocusNode());
     _displaySnackBar(context);

   }

  });
} on FirebaseAuthException catch (e) {
  if (e.code == 'weak-password') {

   _changeTextFieldBoarderColor("red");
   toastMessages('The password provided is too weak.');

   FocusScope.of(context).requestFocus(_signUpPassNode);
  } else if (e.code == 'email-already-in-use') {

   _changeTextFieldBoarderColor("red");
   toastMessages('The account already exists for that email.');
   FocusScope.of(context).requestFocus(_signUpEmailNode);
  } else {
   Future.delayed(Duration(milliseconds: 500)).then((value) {
     _forkProgressIndicator().hide();
     _signUpForkController.dispose();
   }).whenComplete(() {
     toastMessages(e.code);
  });
 }
```

Users can sign in providing their email and password only if their email is verified. After a period of time, the verification email expires, so users can resend the email again through the application. If a user forgets the password, there is the option to reset it by sending an email to their email through the application that includes the steps to achieve that.

5.2.2 Authentication State

This chapter explains the connection between log in screen and main screen. In order for the user to sign in and navigate to the home screen a stream has been used. A stream in dart listens whenever something changes its state. In our case we had to listen to a stream and know whether a user is signed in or signed out from an account. The stream we used listens to authStateChanges() function and has the following form:

```
Stream<User> get _onAuthStateChanged => FirebaseAuth.instance.authStateChanges();
```

This stream is provided to a StreamBuilder widget of Flutter and in any case returns the appropriate screen to user based on the state the user is in.

The Stream Builder is shown below:

```
StreamBuilder(
      stream: _onAuthStateChanged,
      builder: (context, AsyncSnapshot snapshot) {
        if (utils.auth.currentUser != null) {
          if (snapshot.connectionState == ConnectionState.active &&
              utils.auth.currentUser.emailVerified ||
              FirebaseAuth
                  .instance.currentUser.providerData[0].providerId ==
                  'facebook.com') {

            return MainScreen();
          }
          else {
            return Login();
          }
        } else {
          return Login(); } },
   ),
```

Before navigating to the Main Screen, we always check if the email is verified. Using this stream is easier to navigate users to the appropriate screens without needed to perform different checks of user’s current state since the stream triggers immediately after the state changes.

5.2.3 Image Recognition Component

In this section the image processing and the implementation in Flutter will be described. Users are able to access this page by the bottom navigation bar. The first part of the screen is the camera. When a user navigates to the image recognition page, the mobile camera is turned on and ready to capture a picture. Furthermore, a user can choose a picture from the gallery. To achieve that feature, the ''camera'' package is used with all the features provided.

The main three buttons of the screen is the flash button, swap to selfie camera button, choose from gallery button and the capture picture button. Users are able to zoom in and zoom out as well.

5.2.3.1 Flash Button

The flash button consists of three different options. There are flash mode on, auto and flash mode off. The main method that controls each case of the above, is shown below:

```
void onSetFlashModeButtonPressed(FlashMode mode) {
  setFlashMode(mode).then((_) {
    if (mounted) setState(() {});
  });
}
```

The method is called whenever a user choose a flash mode. Depending on this choice, the method changes the flash mode. The widget that contains all the flash modes in the user interface is described below:

```
Widget _flashModeControlRowWidget() {
  return SizeTransition(
    sizeFactor: _flashModeControlRowAnimation,
    child: Center(
      child: Row(
        mainAxisSize: MainAxisSize.min,
        children: [
          Container(
            alignment: Alignment.center,
            decoration: BoxDecoration(
                color: Colors.grey[800].withOpacity(0.5),
                borderRadius: BorderRadius.all(Radius.circular(30))),
            child: ClipRect(
              child: Row(
                mainAxisSize: MainAxisSize.min,
                children: [
                  IconButton(
                    icon: Icon(Icons.flash_off),
                    color: controller?.value?.flashMode == FlashMode.off
                        ? Colors.yellow[300]
                        : Colors.grey[300],
                    onPressed: (){
```

```
                 if(controller!=null){
                   onSetFlashModeButtonPressed(FlashMode.off);
                   onFlashModeButtonPressed();
                   setState(() {
                     icon1=Icon(Icons.flash_off);
                   });
                 }
               }
             ),
             IconButton(
               icon: Icon(Icons.flash_auto),
               color: controller?.value?.flashMode == FlashMode.auto
                   ? Colors.yellow[300]
                   : Colors.grey[300],

               onPressed:(){
                 if(controller!=null){
                   onSetFlashModeButtonPressed(FlashMode.auto);
                   onFlashModeButtonPressed();
                   setState(() {
                     icon1=Icon(Icons.flash_auto);
                   });
                 }
               }
             ),
             IconButton(
               icon: Icon(Icons.flash_on),
               color: controller?.value?.flashMode == FlashMode.always
                   ? Colors.yellow[300]
                   : Colors.grey[300],
               onPressed:(){
                 if(controller!=null) {
                   onSetFlashModeButtonPressed(FlashMode.always);
                   onFlashModeButtonPressed();
                   setState(() {
                     icon1 = Icon(Icons.flash_on);
                   });
                 }
               }
             ),
           ],
         ),
       ),
     ),
   ],
 ),
),
  );
}
```

5.2.3.2 Gallery Button

When users press the gallery button, a navigation to the devices gallery follows. The function that takes action is the openGallery function and its structure is shown below:

```
void openGallery(BuildContext context) async {
  setState(() {
    utils.processing=true;
  });
  final _picker = ImagePicker();
  try {
    var picture = await _picker.getImage(
      source: ImageSource.gallery,);
    File compressedFile = await FlutterNativeImage.compressImage(picture.path, quality: 20,
    );
    Image image = Image.file(File(compressedFile.path));
    var backgroundColor = await functions.getImagePalette(image.image);
    _upperContainerColor=backgroundColor[0];
    if(backgroundColor[1]!=null){
      _bottomContainerColor=backgroundColor[1];
    }else {
      _bottomContainerColor = HSLColor.fromColor(backgroundColor[0])
          .withLightness(0.80)
          .toColor();
    }
    if(picture!=null){
      setState(() {
        imageFile=XFile(picture.path);
      });
    }
    await Future.delayed(Duration(milliseconds: 1000));
    if(mounted){
      setState(() {
        utils.processing=false;
      });
    }

  } catch (e) {
    if(mounted){
      setState(() {
        utils.processing=false;  }); }

 }}
```

After the user picks a photo, a compression of the photo's size and quality is followed in order to load the page faster. As an additional feature and for a better interface, the two main colours of the picture are extracted in order to show them in the top and bottom empty space of the picture, when the preview is shown. To achieve this feature the palette_generator package is used and the two colours extracted are the dominant colour and the light muted colour. The function that accomplishes that is called getImagePalette.

5.2.3.3 Capture Button

When a user captures a picture a method called takePicture takes action and returns the captured file as shown below:

```
Future<XFile> takePicture() async {
  final CameraController cameraController = controller;
  if (cameraController == null || !cameraController.value.isInitialized) {
    showInSnackBar('Error: select a camera first.');
    return null;
  }

  if (cameraController.value.isTakingPicture) {
    // A capture is already pending, do nothing.
    return null;
  }

  try {
    XFile file = await cameraController.takePicture();
    return file;
  } on CameraException catch (e) {
    _showCameraException(e);
    return null;
  }
}
```

After the completion of this function, the preview picture is displayed.

5.2.3.4 Recognize Dish

After a user captures a picture or choose on from gallery, the image recognition follows. In order to recognize a dish from a picture the trained model has to be initiated. The tflite file, produced in the model training process, is saved in the assets directory of the Flutter project. To initiate the file, the loadModel function is called. In addition, another file is needed in order for the model to recognize the name of the predicted class. It is called labels.txt and it contains the name of all 101 classes that the model was trained on. The loading of these two files happens in the initState and is shown below:

```
@override
void initState() {
  super.initState();
  loadModel();

}

loadModel() async{
  Tflite.loadModel(
    model:"assets/food_model.tflite",
    labels: "assets/food_model_labels.txt",
  );
}
```

The next phase is recognizing the dish. The responsible function for that goal is called runModel. By taking the picture file provided and by searching the tflite and labels.txt files, the function returns the predicted class name followed by the confidence (or accuracy) if a dish is found or a message that no dish is found to the user. The structure of this function is shown below:

```
runModel() async{
  setState(() {
    processing=true;
  });
  if(imageFile!=null){
    var recognition = await Tflite.runModelOnImage(
        path: imageFile.path,
        numResults:1,
        threshold: 0.3,
        imageMean: 0,
        imageStd: 255.0,
        asynch: true);
    try{
      if(recognition.isNotEmpty){
        setState(() {
          dishName = recognition[0]["label"][0].toUpperCase() + recognition[0]["label"].substring(1); // recognition[0]["label"]
+ ": " + recognition[0]["confidence"].toStringAsFixed(4);
          accuracy = (recognition[0]["confidence"]).toStringAsFixed(2);
          if(double.parse(accuracy)<0.5){
            setState(() {
              progressBarColor=GFColors.WARNING;
            });
          }else{
            setState(() {
              progressBarColor=GFColors.SUCCESS;
            });
          }
          processing=false;
        });
        _dishNameAnimationController.forward();
      }else{
        functions.showToast("No dish found",2500,ToastGravity.TOP);
        setState(() {
          processing=false;
        });
      }
    }catch(e){
      print(e);
      setState(() {
        processing=false;
        dishName=null;
      });
    }

  }

}
```

The threshold chosen for the model is 0.3, meaning that dishes found with confidence less than this value won't be accepted and no recognized dish will appear. Furthermore, the value chosen for imageMean is 0 and for imageStd is 255. The values selected are precise and emerged from the original trained model. Selecting different values lead to wrong predictions and less accurate.

By tapping the recognized dish, users are navigated to the recipes page. This page contains all the exact or similar recipes of the predicted dish, in a list.

6. Evaluation & Testing

This chapter presents the evaluation and testing of the project. The first two sections present the testing of the functional requirements using the Android framework and the non-functional requirements evaluation respectively. The third section presents the application's UI testing using the Android tools, followed by the application's evaluation. The last section is a summary of all the results of the testing that has taken place.

6.1 Functional Requirements Evaluation

In order to test if the application meets the defined functional requirements in Section 3.1.1, the Android Studio emulator was used as well as some different android mobile devices. We tested all aspects of the application during and after the implementation.

Below the test cases are presented to evaluate the correct functionality of the application.

- Checks that the application allows users to sign up.
- Checks that the application allows users to sign in. At first an indicator is displayed to the user and the user is redirected to the main screen.
- Checks that the application allows users to change the password or resend a verification email after a user sign up.

- Checks that the application allows users to change their name and password after they are logged in.
- Checks that the application is presenting the correct information inside a recipe.
- Checks that the application is uploading data when needed to Firestore.
- Checks that the application is reading the correct data from Shared Preferences, to present users' history.
- Checks that the applications are showing the correct messages to users in all different error cases.
- Checks that the data shown to the users are the correct ones even if the user switch profiles from the same device.
- Checks that the application's data saved in the temporary directory of the device are deleted when users uninstall the application or clear cache.
- Checks that the refreshes in every screen of the application is indeed refreshing the data needed.
- Checks that the application is displaying the correct system notifications to users, according to the app's life cycle state (background, foreground, terminated), in all different cases.
- Checks that in app notifications are displayed to users when the application is in foreground and not displaying system notifications in that case.
- Checks in image recognition page with many different pictures to evaluate the precision of the trained model.

All the functional requirements were met during the implementation and thus all test cases passed. Tests were made during the development of the application as well as when the application was finished.

6.2 Non-functional Requirements Evaluation

In this section, it is checked whether the non-functional requirements of the application which were defined in Section 3.1.2 were successfully fulfilled.

- ➢ The application offers the same user interface in all screens using the same style for menus, buttons and layouts.
- ➢ The application should show clear and detailed notification messages to the user. This was tested by navigating through all the application's screens and using all the functions which have been implemented. For every action a user performs, the system shows a detailed Toast message.
- ➢ The application must have a lack of bugs and must inform the user of any wrong operation. This was tested by doing all the possible mistakes in text fields that user has to fill information such as password or username. Furthermore, running the application and performing all possible actions in every screen, made it clear that the code is properly built.
- ➢ The application should be able to run on all Android devices. This was tested apart from the emulator of Android Studio, in many different devices of different brands.
- ➢ The application should request a password for each user account. Nobody can have access to the main screen of the application without passing through the sign in operation. If the user adds blank data to the password field, the system will display an error recognition message, as it was tested in the test cases in the previous section. The same goes for the sign-up process which displays the same messages when the user leaves the password fields empty.

6.3 Testing The Application's UI

- Checks that the application UI shows the ListView objects in all screens with ListView objects (Recipes).
- Checks that the camera is opening every time a user switch to the image recognition tab.
- Check that the application UI shows all buttons correctly in all screens.
- Checks that all screens have lack of crashing UI errors.
- Checks that the application UI shows the error toast messages correctly.
- Checks that the UI is showing the correct information in recipes including the recipe information, the rating of the recipe and the favorite button which is filled if a user has added the recipe to their favorites.

All the User Interface test cases completed with almost no errors. Some errors were reported by the users using the application but were minor problems, such us position of toast message display position in the screen, but they were fixed immediately when reported.

6.4 Loading TFlite In Flutter

After training our image recognition model in Google Colab, the next step was the implementation in Flutter. There were many attempts to load the model in Dart but they all failed. After trying many different things, we concluded that there was a problem in the training. During the first training, the input shape in the InceptionV3 algorithm wasn't specified. With the absence of input shape, the model was using the default one. The input shape determines the input picture dimensions which in our case were 299x299 pixels in RGB (3 channels). The code sample is shown below:

```
inception = InceptionV3(weights='imagenet', include_top=False , input_shape=(299,299,3))
```

With the addition of the input shape and by retraining the model, the problem of loading the model in Dart was solved.

6.5 Tflite Size

As described in a previous chapter, after the training of the model, a conversion of the file to a tflite file had to be done in order to read the model in Flutter. The original size of the tflite file was 120 Mb, a very big number. The file had to be compressed. The main reason is that the model will be a part of the application, which means that by downloading the application a user will download the model as well. The size of the application is 80 Mb, so with the addition of the models it would have approximately 200 Mb size, which is a lot for an application. The solution was provided by TensorFlow. During the conversion of the model to a tflite file, optimizations can be used in order to control the size of the tflite file. The optimizations using TensorFlow are shown below:

```
converter.allow_custom_ops = True

converter.optimizations = [tf.lite.Optimize.DEFAULT]
converter.target_spec.supported_ops=[tf.lite.OpsSet.TFLITE_BUILTINS]
```

After this addition, the conversion process completed as shown in the code sample below:

```
import tensorflow as tf

converter = tf.lite.TFLiteConverter.from_saved_model(r"path-of-the-saved-model")

converter.allow_custom_ops = True

converter.optimizations = [tf.lite.Optimize.DEFAULT]

converter.target_spec.supported_ops=[tf.lite.OpsSet.TFLITE_BUILTINS]

tflite_model = converter.convert()

with open(r'/content/drive/MyDrive/FoodRecognition/FoodRecognitionExp/modelQuantized.tflite', 'wb') as f:

  f.write(tflite_model)
```

After the conversion the tflite size fell from 120 Mb to 28 Mb, a much more reasonable number to add in the application size.

6.6 Bad Internet Connection

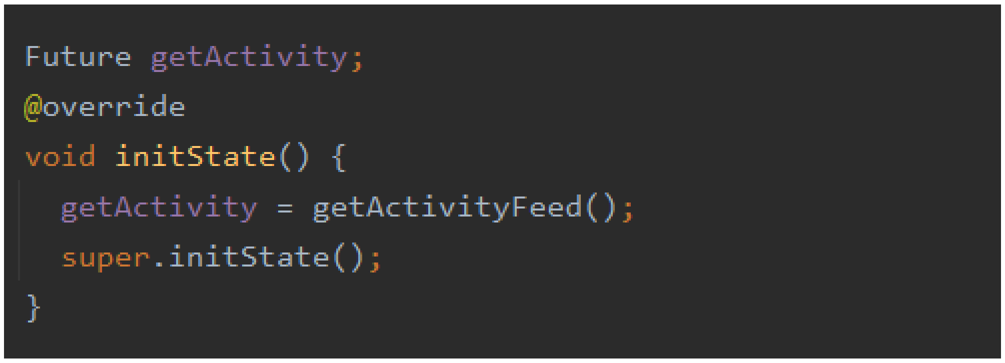
A problem with the login occurred when users were on an unstable internet connection. When the user is registering through sign in with google, Facebook or email, after they have completed the registration and they return to the app, the state does not update properly. When this problem happened, the users were stuck seeing a progress indicator. There is no such way to avoid this problem so the user must find a way to improve the internet connection and try to sign in again to the application. Unfortunately, these kinds of problems are mostly concern the user and not the developer of an application. The implementation that is done, is the use of a package called connectivity which tracks whether a user is connected or not to an internet. When the change of the connection state is happening, our application is displaying a message and informs the user if there is an internet connection or not.

Apart from the sign in process with a bad internet connection, there is an implementation of what should happen when no connection is available to almost every widget in code. That means that if there is no connection, the users Posts for example will display a message in the middle saying ‘’No internet Connection’’ instead of the image or video, but the user in any case will not face a red error screen.

6.7 Database Reading Problems

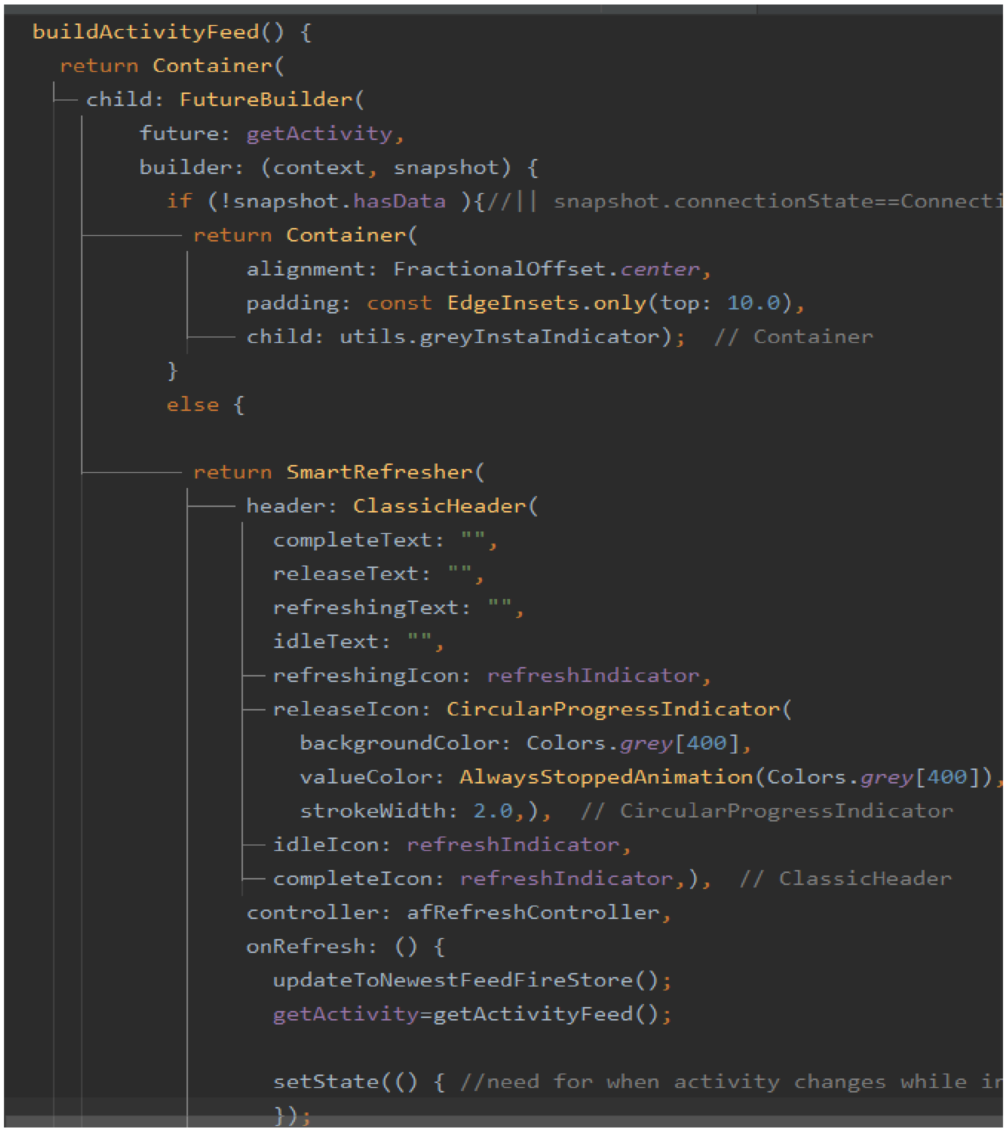
During the application development we found out that the number of reads made from the database were unexpectedly high. After a period of time testing the application, we found that the source of this problem was the use of a widget in Flutter called FutureBuilder. What FutureBuilder does is that while building a page, for example the Comment Page, it waits for its future function to be completed meaning that it will wait for the function to read all the data from Firestore and display them after. The combined use of a FutureBuilder with a setState function (that refreshes the UI if its needed) causes the future function of the FutureBuilder to run again meaning more reads from Firestore.

To avoid these kinds of problems, our first action was to initialize the future function inside the initState function of the class. Doing that decreased the number of reads made but didn't entirely solve this problem unfortunately. This is presented in Figure 4.1 and Figure 4.2.



```
Future getActivity;
@override
void initState() {
  getActivity = getActivityFeed();
  super.initState();
}
```

Figure 5.1 Future in initState Function



```
buildActivityFeed() {
  return Container(
    child: FutureBuilder(
        future: getActivity,
        builder: (context, snapshot) {
          if (!snapshot.hasData ){//|| snapshot.connectionState==Connecti
            return Container(
                alignment: FractionalOffset.center,
                padding: const EdgeInsets.only(top: 10.0),
                child: utils.greyInstaIndicator);  // Container
          }
          else {

            return SmartRefresher(
                header: ClassicHeader(
                  completeText: "",
                  releaseText: "",
                  refreshingText: "",
                  idleText: "",
                  refreshingIcon: refreshIndicator,
                  releaseIcon: CircularProgressIndicator(
                    backgroundColor: Colors.grey[400],
                    valueColor: AlwaysStoppedAnimation(Colors.grey[400]),
                    strokeWidth: 2.0,),  // CircularProgressIndicator
                  idleIcon: refreshIndicator,
                  completeIcon: refreshIndicator,),  // ClassicHeader
                controller: afRefreshController,
                onRefresh: () {
                  updateToNewestFeedFireStore();
                  getActivity=getActivityFeed();

                  setState(() { //need for when activity changes while ir
                  });
```

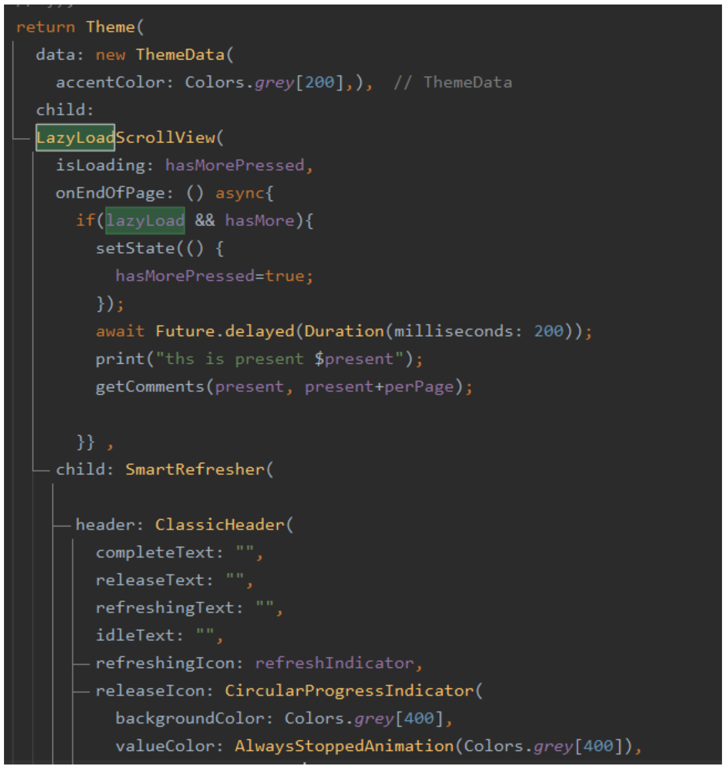
Figure 5.2 FutureBuilder with future Function

The main reason this had to be fixed was that if a lot of users were using the application the number of reads from the database would be a very big number and that would make the cost go up dramatically.

The more drastic action we made, was the implementation of Paginator, meaning we didn't bring all the results when needed but we brought batches of results, for example by 20, when the users scroll to the end of a list. This method implemented in many screens such as the Post screen, Comment screen, Recipes screen and User Profile screen.

In order to implement the pagination of lists in a ListView widget in Flutter, we used a library called Lazy Load Scrollview. The library allows the use of a LazyLoadScrollview widget that ‘’does’’ something when the users scroll to the end of a list. Using variables that are incremented when user scrolls to the end of the list, we managed to accomplish what we wanted. In case of refreshing the list, the variables value is returning to its original desired value.

As shown in Figure 4.3, LazyLoadScrollView widget has and attribute called onEndOfPage. This attribute allows us to implement the code for what will happen when a user scroll to the end of the list. The two variables present and perPage are the variables that we increment to get the next batch of the results to the list.



```
return Theme(
  data: new ThemeData(
    accentColor: Colors.grey[200],),  // ThemeData
  child:
  LazyLoadScrollView(
    isLoading: hasMorePressed,
    onEndOfPage: () async{
      if(lazyLoad && hasMore){
        setState(() {
          hasMorePressed=true;
        });
        await Future.delayed(Duration(milliseconds: 200));
        print("ths is present $present");
        getComments(present, present+perPage);

      }} ,
    child: SmartRefresher(

      header: ClassicHeader(
        completeText: "",
        releaseText: "",
        refreshingText: "",
        idleText: "",
        refreshingIcon: refreshIndicator,
        releaseIcon: CircularProgressIndicator(
          backgroundColor: Colors.grey[400],
          valueColor: AlwaysStoppedAnimation(Colors.grey[400]),
```

Figure 5.3 Lazy Load ScrollView widget

6.8 Summary

The goal of this Chapter was to present all the tests and evaluations that took place during the implementation phase and afterwards to evaluate the correct functionality of the application. All test cases for the functional requirements were passed which means that the application has no functional error. The non-functional requirements were met too. In order to test the UI beside on our own by the use of Android Studio emulator, we asked a group of people to use the application for a period of time and report any problem they meet during that period. Last but not least, in order to solve the increased number of reads from Firestore we implemented pagination to almost all the lists used in the application. There will always be some room for errors or bugs in the app but they will be resolved as long as they appear.

7. Conclusion

This chapter summarizes the work done in this project, followed by the project's contribution and future work.

7.1 Work Summary

The goal of the project was to create a social media mobile application called Cook Eat. In addition, the use of Firebase and its services was another goal for the project. Solving common or more complex development problems and trying to make a friendly and interesting user interface brought us closer to that goal. The application, as mentioned in chapter 5, has passed almost all the tests we have implemented as a result we hope to give an excellent experience to our users.

7.2 Project Contribution

The contribution to the mobile developers' community is described in this section. The contribution is addressed below:

- Design principles and patterns of mobile interface design were analyzed. Instructions were given, which can be followed by new mobile developers for designing and developing good applications. This analysis may also help developers understand and consider a few important things when they want to design a new mobile application or transfer a full-sized computer application to a mobile environment.
- Tactics for solving common problems in Android application development to help new developers were provided.
- A fully functional Android application was created. The main advantages of creating this application are:
  - It is designed to recognize pictures of food and give users access to instructions of recipes they want to cook.
  - As it is an Android application, it can be installed for free in all Android devices (mobile phones and tablets), by being shared in the Google Store.
  - The source can be used by other developers to extend the current functions or add new ones that will enhance design of social media applications and tactics with Firebase.
  - Well design and strict login or sign up development can contribute to other developers future work in applications.

Evaluation of the application was made. The evaluation was based on unit testing for the functional requirements and UI. We took into consideration the feedback of a group of people using the applications for a period of time.

7.3 Future Work

This section discusses the possible improvements that can be made in the future to improve the application and add new functions.

→ **Improvement in Navigation.** The GUI of the application offers a minimal and easy navigation through the application's screens. As a result, the user needs to go to the main screen to navigate from the main to other screens. To avoid this, future developers can implement a navigation drawer. The navigation drawer is a panel that transitions in from the left edge of the screen and displays the application's main navigation options. It has the ability to access any top level content from anywhere in the application, no matter how deep the level a user is in.

→ **Improvement in Sign In method.** Currently, the check if the user is logged in or not is done through a StreamBuilder widget. There are some problems using a stream builder, especially when the internet connection of the user is bad. To avoid this, developers can implement many different ways such as SharedPrederences, Provider widget and more.

→ **Add charts.** As a feature, developers can add charts, giving information to the users such as average time users used the application the past week or how often the upload posts in the app and more.

→ **Implement data analysis.** Information about a users average age or gender or preferred recipes and categories of food could be an excellent work for future developers. The implementation can be done by developers by adding a form for users to fill, then extract the information and analyze it. A recommendation system of recipes can be added as well, by analyzing users preferences and most visited recipes or recipes they add to their favorite collection. For this to happen, developers must add extra collections to Firestore and save all the important information needed for such a system.

→ **Use more Cloud Functions.** Many transactions with Firestore can be done with Cloud Functions. Reading data from Firestore and displaying that data to users, even with the use of paginators, is sometimes a heavy process for the mobile device. As a future work, developers could use Cloud Functions to retrieve data for recipes, in-app notifications, personal posts, followers and more.

→ **Add Chat.** This feature is pretty common to most social media applications. Future developers could add messaging for more socialization between users. Chat rooms, for a group of people or one to one conversation, could be implemented giving a better experience to users.

→ **Identify inappropriate content**. The image or video of a post can have inappropriate content because there is no check that is happening now. With the help of Machine Learning, a model that identifies this kind of content can be built in order to check the content of each post or comment, making it impossible for a user to make an upload with inappropriate content in the application.

→ **More exciting features.** These kinds of applications that are referring to socializing between users, have more room for many features and trends that a developer can add to the application. Instagram stories, for example, was something new and exciting for users. As an interesting feature, developers could add videos in recipes so users can watch the cooking process and make it easier for them to cook the recipe. Another feature could be giving the option to users to upload their own recipes in FireStore and other users could have access to these recipes, as well as rate the recipe or add it to their favorite collection.

7.4 Concluding Remarks

Having mentioned both the theoretical and the practical aspects of mobile application development, as well as its primary uses and benefits, we can provide an overall account of our study. To begin with, in our project we analyzed the development process followed for Android applications, as well as the design principles, patterns and tactics which are useful for mobile application development. Having taken all these factors into consideration, we put them into practice and created a social media with image recognition Android application called CookEat. To conclude, we believe that, with further improvement from future developers, the already useful application we created could become as well known as other applications that offers even more features to users.

References

1. Mayfield, Antony. "What is social media." (2008).

2. Bradley P. Be where the conversations are: The critical importance of social media. Business Information Review. 2010 Dec;27(4):248-52.

3. Vinerean S. Importance of strategic social media marketing.

4. Miller D, Sinanan J, Wang X, McDonald T, Haynes N, Costa E, Spyer J, Venkatraman S, Nicolescu R. How the world changed social media. UCL press; 2016.

5. Perrin A. Social media usage. Pew research center. 2015 Oct 8;125:52-68.

6. Ruths D, Pfeffer J. Social media for large studies of behavior. Science. 2014 Nov 28;346(6213):1063-4.

7. Wilson RE, Gosling SD, Graham LT. A review of Facebook research in the social sciences. Perspectives on psychological science. 2012 May;7(3):203-20.

8. Miller D. Tales from facebook. Polity; 2011 Apr 11.

9. Kwak H, Lee C, Park H, Moon S. What is Twitter, a social network or a news media?. InProceedings of the 19th international conference on World wide web 2010 Apr 26 (pp. 591-600).

10. Myers SA, Sharma A, Gupta P, Lin J. Information network or social network? The structure of the Twitter follow graph. InProceedings of the 23rd International Conference on World Wide Web 2014 Apr 7 (pp. 493-498).

11. Skeels MM, Grudin J. When social networks cross boundaries: a case study of workplace use of facebook and linkedin. InProceedings of the ACM 2009 international conference on Supporting group work 2009 May 10 (pp. 95-104).

12. Sumbaly R, Kreps J, Shah S. The big data ecosystem at linkedin. InProceedings of the 2013 acm sigmod international conference on management of data 2013 Jun 22 (pp. 1125-1134).

13. Miles J. Instagram power. McGraw-Hill Publishing; 2013.

14. Moreau E. What is Instagram, anyway. Here's what Instagram is all about and how people are using it [online]. 2018.

15. Mohri, M., Rostamizadeh, A., & Talwalkar, A. (2018). *Foundations of machine learning*. MIT press.

16. Petrou, M. M., & Petrou, C. (2010). *Image processing: the fundamentals*. John Wiley & Sons.

17. Hastie, T., Tibshirani, R., & Friedman, J. (2009). Unsupervised learning. In *The elements of statistical learning* (pp. 485-585). Springer, New York, NY.

18. Cunningham, P., Cord, M., & Delany, S. J. (2008). Supervised learning. In *Machine learning techniques for multimedia* (pp. 21-49). Springer, Berlin, Heidelberg.

19. Chwif, L., Barretto, M. R. P., & Paul, R. J. (2000, December). On simulation model complexity. In *2000 winter simulation conference proceedings (Cat. No. 00CH37165)* (Vol. 1, pp. 449-455). IEEE.

20. Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017, August). Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)* (pp. 1-6). Ieee.

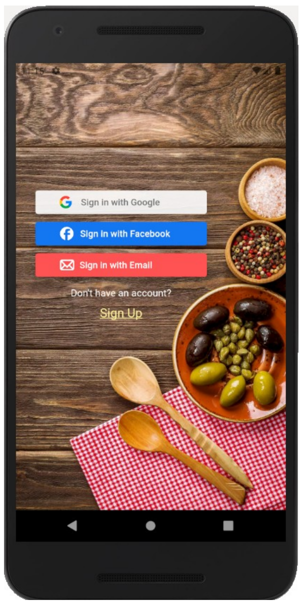
21. Brownlee, J. (2016). *Deep learning with Python: develop deep learning models on Theano and TensorFlow using Keras*. Machine Learning Mastery.

22. Bahrampour, S., Ramakrishnan, N., Schott, L., & Shah, M. (2016). Comparative study of caffe, neon, theano, and torch for deep learning.

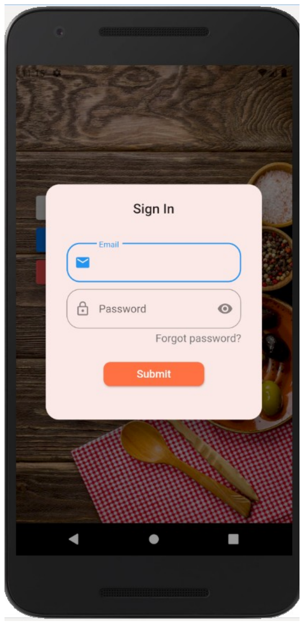
Annex - User Interface

This chapter will provide all different screens of the application and some key widgets that used in Flutter code.

Users are able to sign in with three different methods, Google, Facebook and Email/Password.



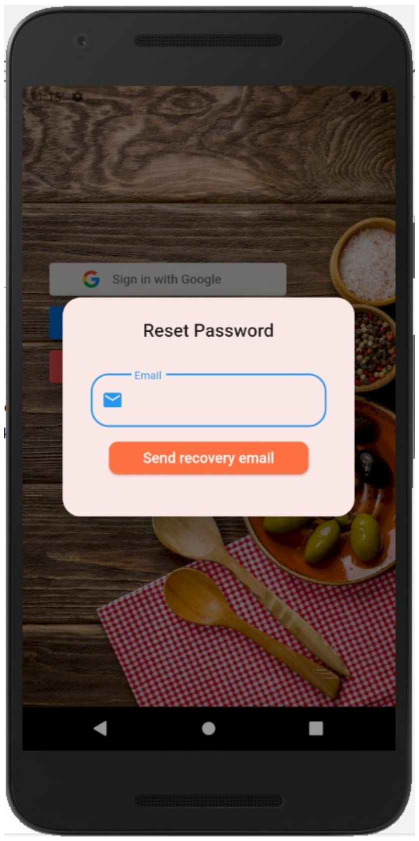
1. Login Screen



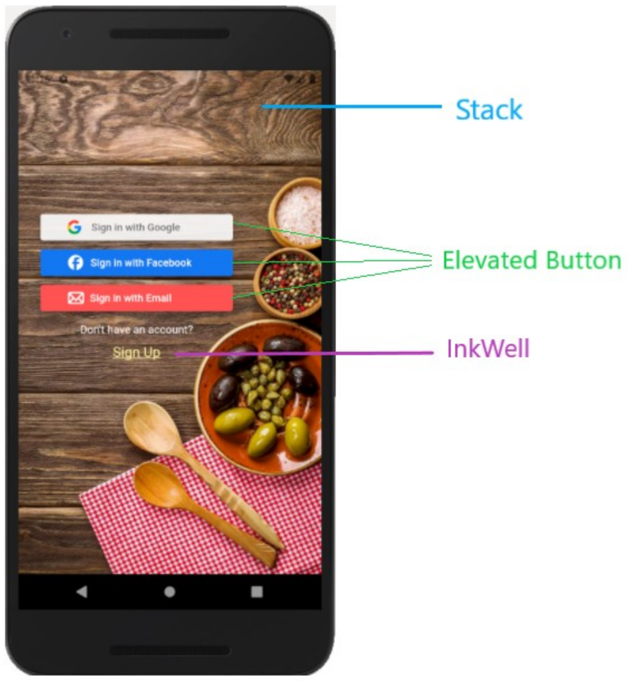
2. Sign In with Email

In Sign in with Email dialog, users are requested to fill their email and provide the password.

Users are able to reset their password in case they forgot it by entering their email address of the email they want to recover the password.

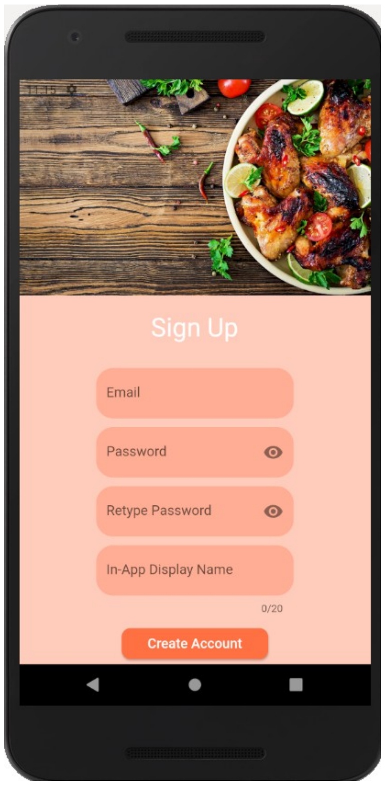


3.Reset Password

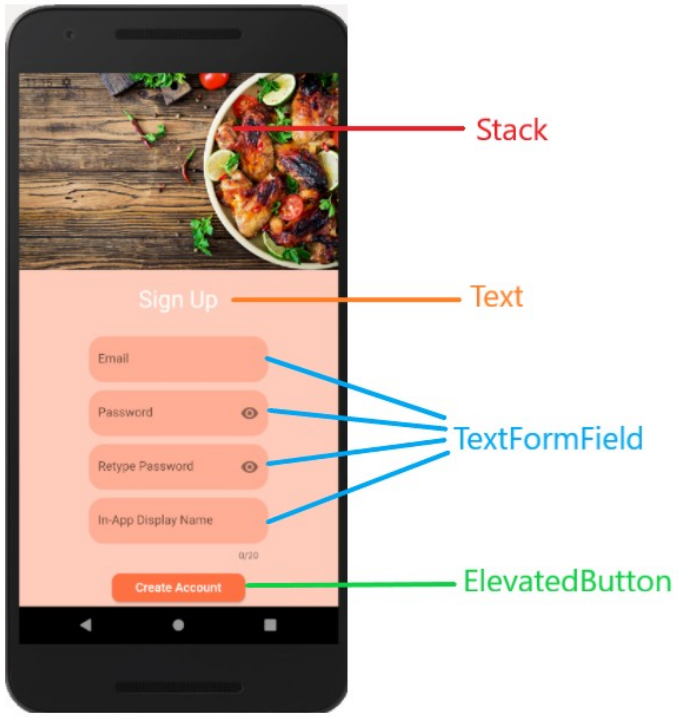


4. Login Screen Widgets

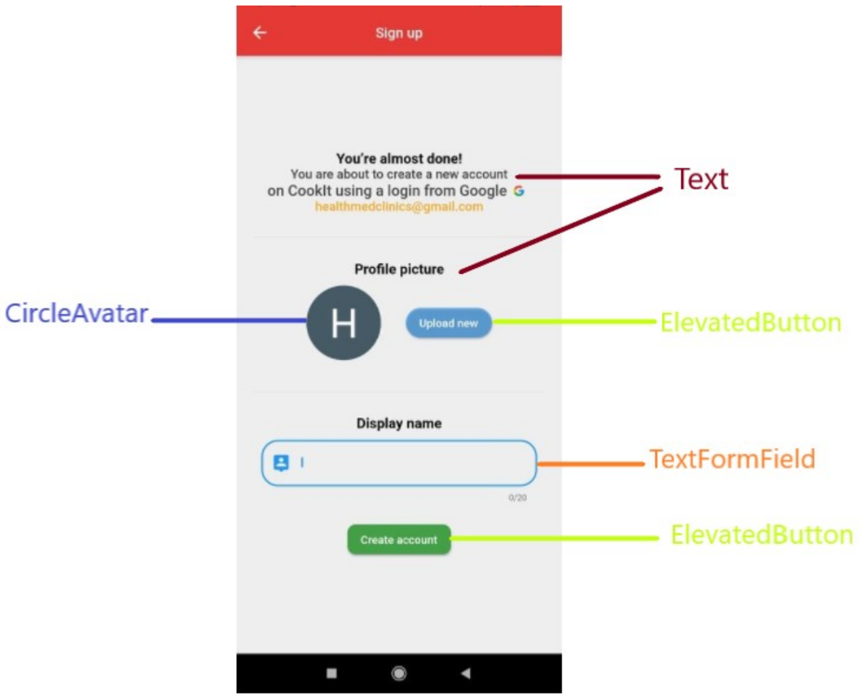
In sign up screens users are requested to fill a form and provide data such as their email, password and the display name of their choice.



5. Sign Up Screen



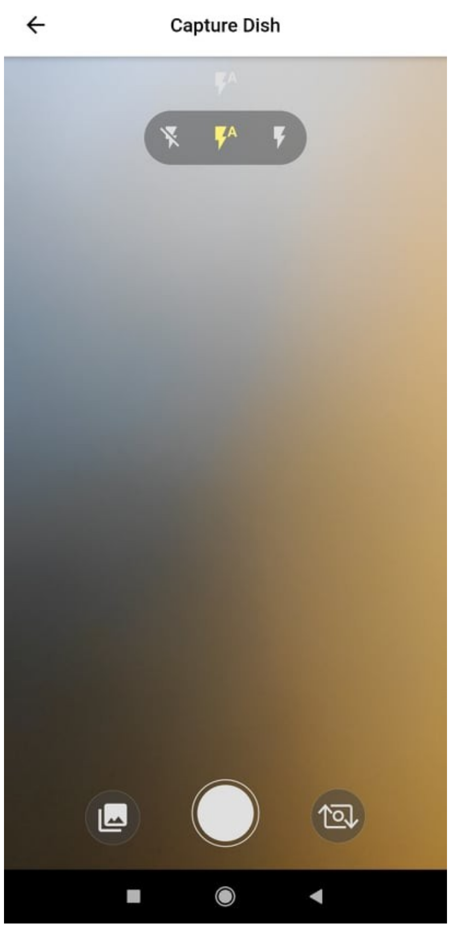
6. Sign Up Screen Widgets



You're almost done!
You are about to create a new account
on CookIt using a login from Google
healthmedclinics@gmail.com

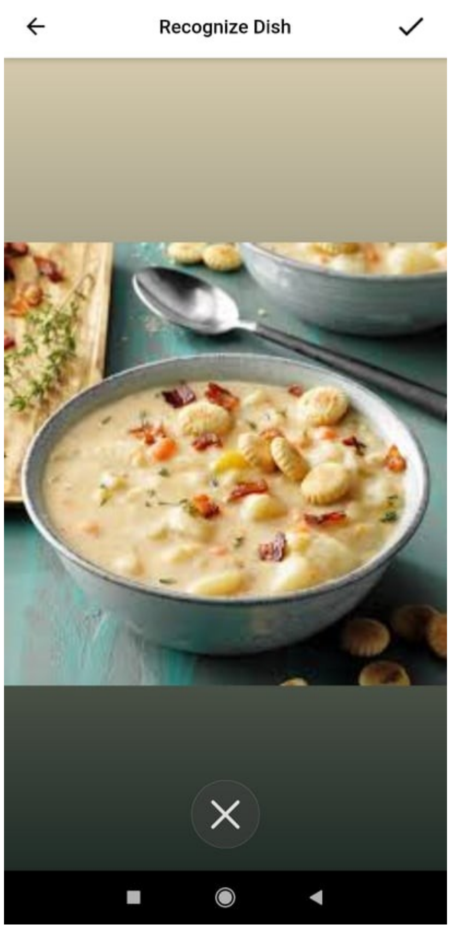
7. Google Sign Up Screen

In image recognition page users are able to take a picture with camera or choose one from the device gallery.



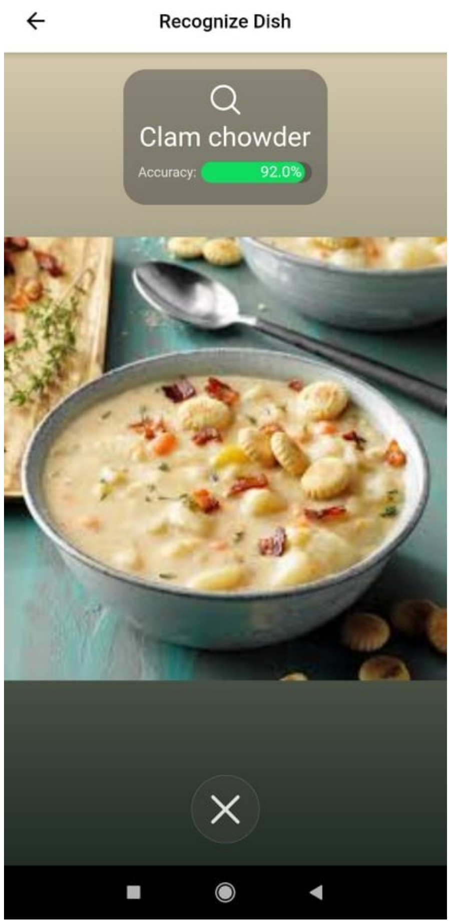
8. Image Recognition Screen

After choosing a photo, it appears on the screen.



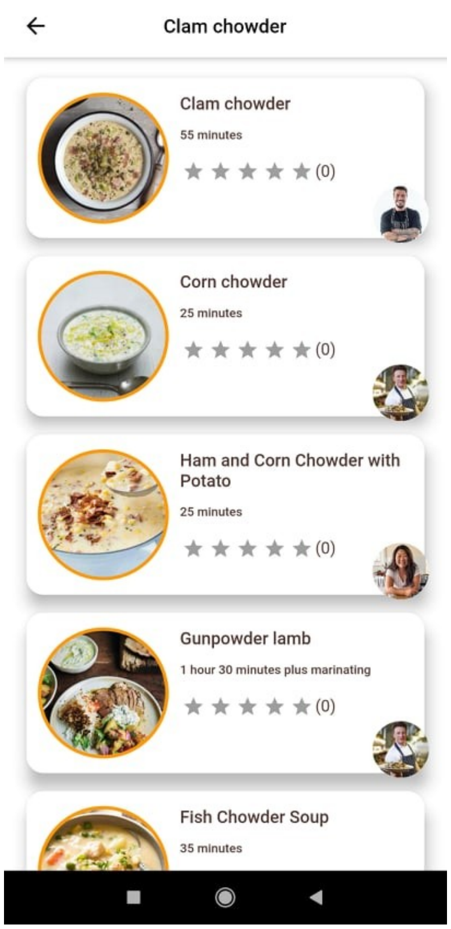
9. Image Recognition Screen

By pressing the done button, a dialog appears with the description of the recognized dish and its accuracy.



10. Recognized Dish

By pressing the dialog, users are navigated to a screen with a list of recipes similar to the one recognized.



11. List of similar recipes