

IM3080 Design and Innovation Project

(AY2020/21 Semester 1)

Project Report

Title: gogoGoals - A Goal Tracking Application

Github:

<https://github.com/NTU-AY2020-DIP-Group-8/gogogoals>

(Application)

<https://github.com/NTU-AY2020-DIP-Group-8/api> **(API)**

Submitted by: Group 8

Ye Jieyi

Tan Kaiwei

Rao Yuchen

Nur Khalisah Binte A Noh

Loh Yan Wen

Lkhagvadorj Dulguun

Li Mingcong

Keerthana Prabakar

Hshieh Yeu Chiann

Elizabeth Mary Lau Jia En

Chen Liangyu

Supervisor: Assoc Prof Teh Kah Chan

Table of Content

Background and Motivation	4
Background	4
Motivation	4
Objective	5
Review of Literature/Technology	6
Current Products	6
Programming Language and Framework	8
Database	8
API Resources	9
Design and Implementation	11
Design Consideration / Choice of components	11
Final Design (with block diagrams & UI)	15
ER & Class Diagrams	15
Site Map & Page Design	16
Implementation	22
Implementation of Key Features	22
Implementation of Database	31
Discussion	32
Design and Implementation	35
Conclusion	35
Recommendation for Future Works	35
References	36
Appendices:	37

A. Bill of Materials	37
B. Design Diagrams	37
C. User Guide	40
Getting Started	40
Prerequisites	40
Android	
iOS	
Features	41
Intelligent Recommendation	
Add Multiple Todos in One Go	
Machine Learning OCR Algorithm for Handwritten Note Conversion	
Time Bound	
Progress Bar	
Highly Customisable UI	
Contribute to gogoGoals	44
D. Maintenance Guide	44
E. How-To Guides	44
Intelligent Recommendation	
Add Multiple Todos in One Go	
Machine Learning OCR Algorithm for Handwritten Note Conversion	
Time Bound	
Progress Bar	
Highly Customisable UI	
Contribute to gogoGoals	
F. Source Code	45
G. Others	46

1. Background and Motivation

1.1. Background

Personal productivity determines how many goals one can achieve within a time bound successfully. The level of personal productivity also has a positive correlation to one's academic, professional and personal success (Erik et al., 2014). The often-recommended method to increase productivity is by goal setting and achievement (MindTools, n.d.). In addition, a study has also shown that frequently monitoring progress toward goals increases the chance of success, and even more so if you report your progress publicly or physically record it (ScienceDaily, 2015). With the advancement of technologies, "paper-written" To-Do-Lists or goal settings have been transformed into many goal tracking mobile applications (apps) that have been marketed to improve productivity. Data from 42 matters, which is an app market statistics provider, have shown that, there are more than half a million goal tracking mobile app downloads in google play station and twenty-five thousand monthly downloads in iTunes app store, this is a clear indicator that goal tracking mobile app has a high market demand.

1.2. Motivation

There are many similar goal tracking Apps found in Google Play Store or App Store that met the basic need of noting down goals and setting a reminder, but seldom do they pay attention to other factors that may affect the level of commitment in accomplishing these goals, especially procrastination — a significant enemy to productivity (Erik et al., 2014). In addition, there are little studies done on factors that influence the effectiveness of goal tracking mobile apps on productivity. Therefore, this is a gap that demands further studies and developments.

2. Objective

The aim of this project was to determine the most significant factors in a mobile tracking app that may enhance users' productivity in terms of the number of goals achieved within a time bound. Followed by developing a mobile goal tracking app that focuses on increasing personal productivity based on these factors. We will begin by examining the effect of higher customizability within the app such as ability to change icons and their colours; the effect of novel features in a mobile tracking app such as intelligent goal recommendations based on user-input and category by integrating external application programming interfaces (APIs); as well as the effectiveness of implementing S.M.A.R.T. strategy in a mobile app. S.M.A.R.T. stands for Specific, Measurable, Achievable, Relevant, and Time-bound that has been adopted by many successful individuals to come up with concrete plans in achieving measurable goals (MindTools, n.d.).

In helping users to visualise their progress at a glance and keep them on track, there will be a progress bar indicating the percentage of goals completed under each category. The effect of having a progress bar on improving productivity will also be assessed. Insights from these experimental results will help developers to decide on the most suitable features to optimise users' productivity level in a mobile goal tracking app.

The software development platform will be Flutter because it provides flexibility of operating in various Operating Systems — Android, iOS, Linux, Mac, Windows, Google Fuchsia and the web from the same codebase.

3. Review of Literature/Technology

3.1. Current Products

Strides is an application that tracks all your Goals & Habits. With Strides, you can track anything - good or bad habits and SMART goals - with reminders to hold you accountable and charts to keep you motivated. Recommended in the New York Times, Strides has been called “the most comprehensive, user-friendly, beautifully designed goal tracking app available.”

Track anything you want with four types:

1. Track good or bad Habits with a streak calendar.
2. Reach your Target goal value by a specific date.
3. Daily, weekly, monthly, yearly or rolling Average.
4. Complete a Project on time with milestones.



Figure 1: Strides

Microsoft To Do (previously styled as Microsoft To-Do) is a cloud-based task management application. It allows users to manage their tasks from a smartphone, tablet and computer. The technology is produced by the team behind Wunderlist, which was acquired by Microsoft, and the stand-alone apps feed into the existing Tasks feature of the Outlook product range.

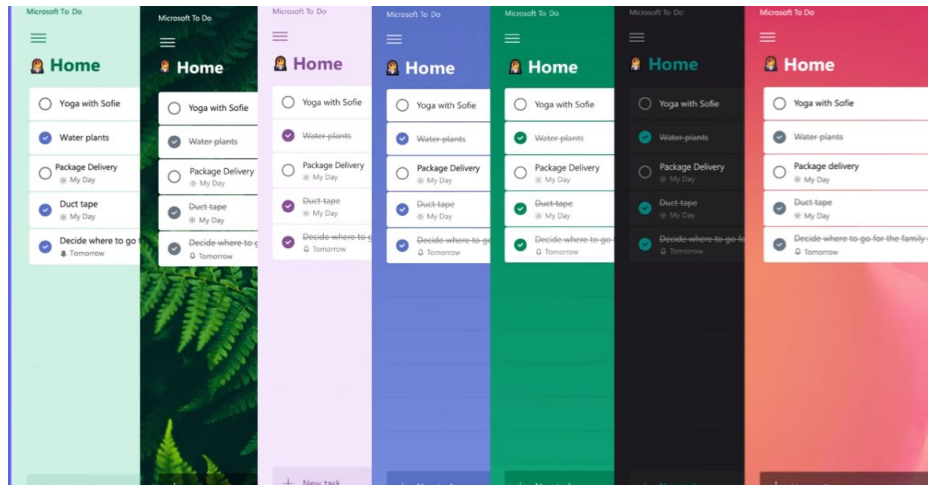


Figure 2: Microsoft to do

Habitica (formerly HabitRPG) is a goal-tracking app that “seeks to gamify your life”, providing users with in-game rewards and punishments that motivates them to stick to their goals and check off tasks. As a role playing game, users will meet with quests to be completed through the completion of goals of their own, while inability to complete will cause them with penalties along the quest.

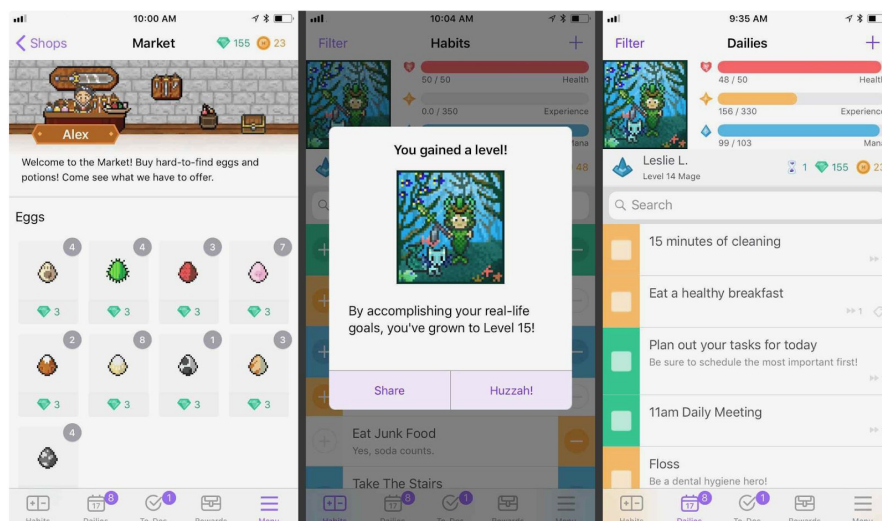


Figure 3: Habitica

3.2. Programming Language and Framework

For the mobile application, the Programming Language we are using is dart and the Framework that we have decided on is Flutter, a free open-source mobile UI (User Interface) Framework created by Google. It allows us to create a native mobile application with only one codebase, which means that we can use one programming language and one codebase to develop two different apps for iOS and Android.

Flutter consists of an SDK (Software Development Kit), a collection of tools that will help develop the application and it includes tools to compile our codes into native machine code for iOS as well as Android. It also consists of a Framework, a UI Library based on widgets, a collection of reusable UI elements like buttons, text inputs and more that we can personalize for our own needs.

To develop with Flutter, we will use a Programming Language called Dart. Dart was also created by Google and has improved a lot over these past years. Dart focuses on front-end development, allowing us to create mobile and web applications.

We have also coded our own API (this API handles random recommendations for categories we didn't find relevant apis for) using Node.js (an open-source, cross-platform, back-end, JavaScript runtime environment that executes JavaScript code outside a web browser) and it was published using Google cloud function.

3.3. Database

Databases are a collection of organized information that can be easily accessed, managed and updated. Database system is very important as it stores, organizes and manages a large amount of information within a single software application, hence we decided to use Firebase.

Firebase is a very powerful platform that developers can use to build applications quickly. Flutter and Firebase work hand-in-hand to help build mobile applications in record time. Firebase gives access to backend services for mobile applications which includes authentication, storage, database and hosting – without maintaining our own servers.

3.4. API Resources

We have incorporated a number of APIs in our codes, to provide users with ideas when they try to find tasks to do, or recognize users' handwritten to-do notes. These recommendation APIs are used based on our broad categories, which taps on the respective databases, allowing users to choose related tasks. The APIs used are as listed below.

Recommendation:

Knowledge category:

- Course recommendation: <https://build.coursera.org/app-platform/catalog/>

Coursera is an online learning platform that provides a vast variation of courses for anyone to take on. They provide courses of different levels, ranging from free courses to university degrees, available for users to choose and learn from.

- Book recommendation: <https://developers.google.com/books/docs/overview>

Google books has an extensive collection of published books, with free previews of the content as well as information needed to find and purchase the book. With a search engine attached, users can easily look for books based on keywords.

Meal category:

- <https://spoonacular.com/food-api/docs>

Spoonacular is a food management system that provides users with everything they need in relation to eating, ranging from meal planners to calorie counters. Having a food search engine, users can source for recipes and store bought products in order to reach their nutrition goals.

Travel category:

- <https://www.triposo.com/api/>

Triposo is a social travel platform that gives users destination recommendations based on their trip information. Users can also book their hotels, sights, activities and restaurants using the app itself.

Synonym matching:

- <https://words.bighugelabs.com/site/api>

Big Huge Thesaurus API is based on data from the Princeton University WordNet database, the Carnegie Mellon Pronouncing Dictionary, and other sources. It serves as a platform for generating synonyms and antonyms for developers.

Text recognition:

- <https://developers.google.com/ml-kit/vision/text-recognition>

ML Kit is Google's machine learning toolkit for mobile developers. With ML Kit's text recognition APIs can recognize printed or handwritten text in any Latin-based character set. They can also be used to automate data-entry tasks such as processing credit cards, receipts, and business cards.

4. Design and Implementation

4.1. Design Consideration / Choice of components

The User Experience (UX) and User Interface (UI) are critical to any software product. UX focuses on the actual functionality and usability while UI refers to the overall style and design of the app. The design considerations of our app includes:

- Making actions reversible: Users are able to undo whatever they are doing or even be able to edit after confirming their selection. This allows more room for errors, allowing the user to use the app freely without the constant fear of failure.
- Creating interfaces that are easy to navigate: Users are able to navigate through the app easier with the help of visual cues like coloured buttons and highlighted words that will cue their next step in the app.
- Well crafted error messages: Making errors is inevitable, with our well crafted error messages, it allows users to know exactly where the error is and what steps must be taken next to correct it.
- Fitt's Law to interactive elements: Fitts Law states that the time to acquire a target is a function of the distance to and size of the target, meaning larger buttons for important functions like what we have done.
- Visual consistency: We used the same colours, fonts and icons for every page of the app allows visual consistency, making it less confusing for the user.

Initial UI Drafts

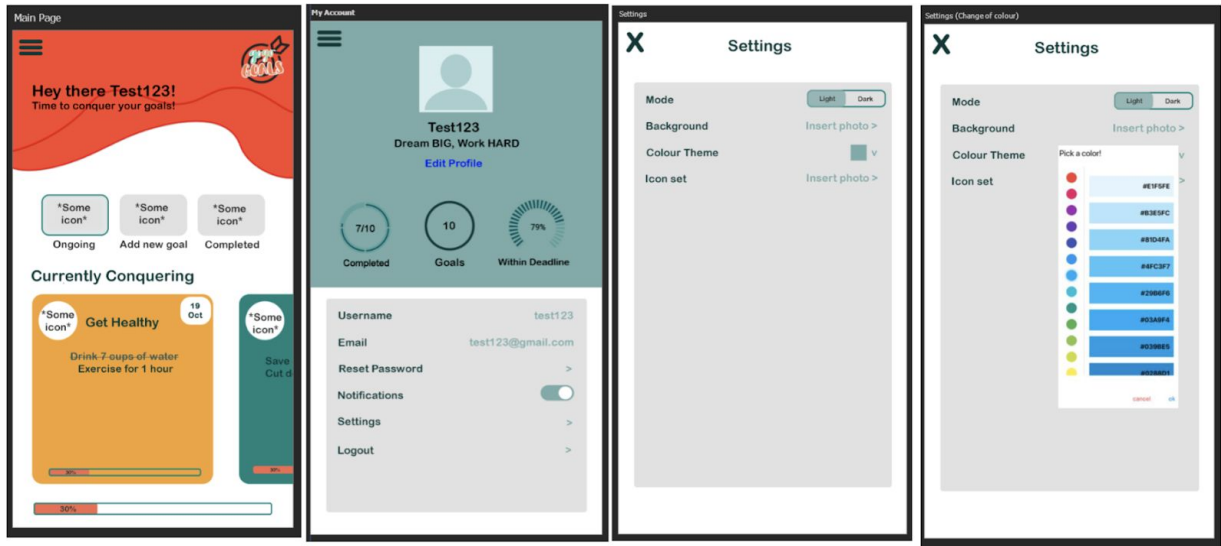


Figure 4: UI Design 1

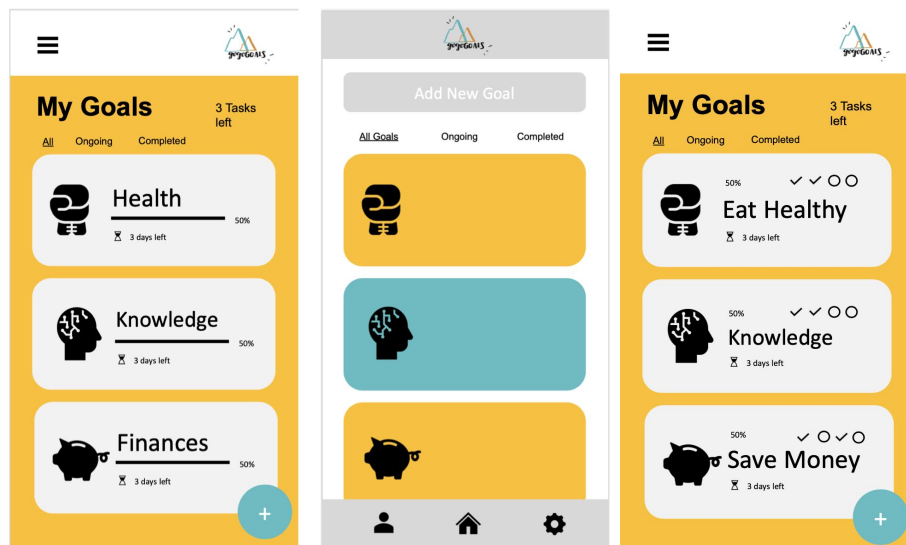


Figure 5: UI Design 2

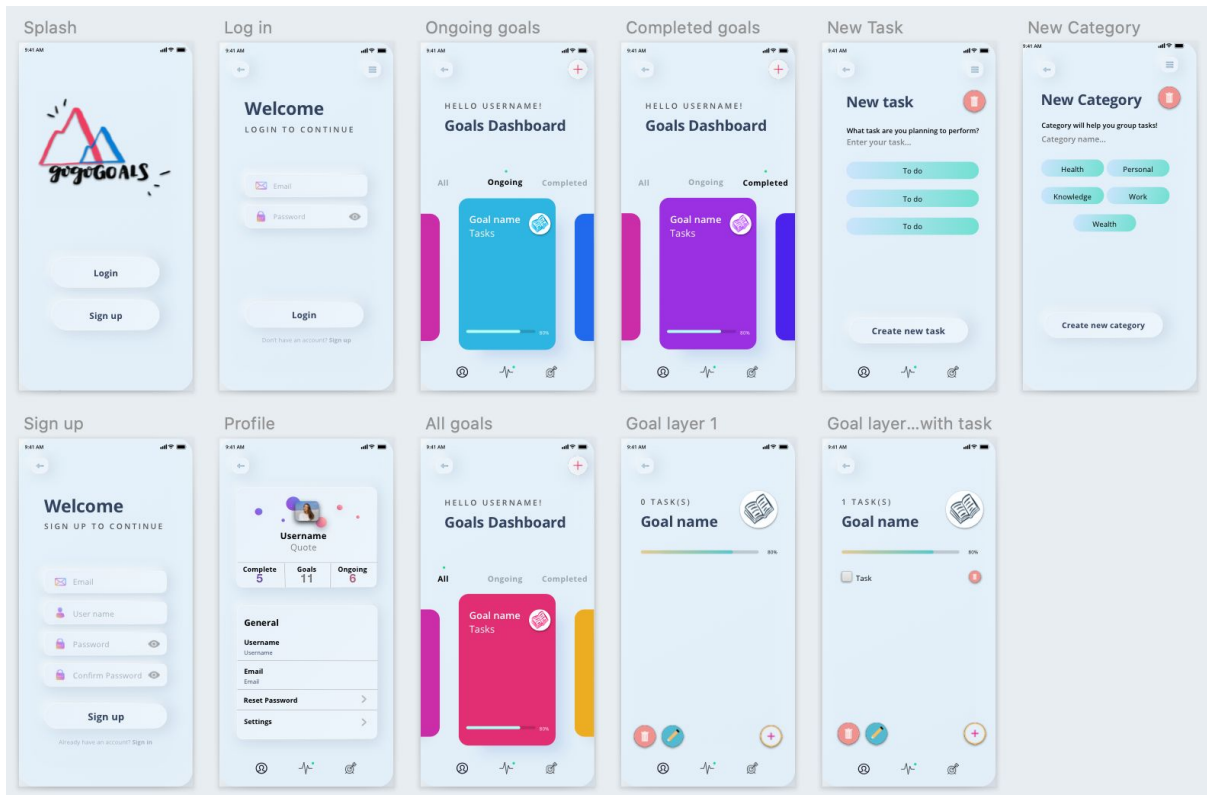


Figure 6: UI Design 3

With our key design components decided, we came up with a number of User Interface prototypes using Photoshop and Sketch. UI design 1 (Figure 4) gave us the main skeleton of our application's user interface. UI design 2 (Figure 5) gave us the layout for the homepage. UI design 3 (Figure 6) gave us the wireframe and overall elements structure for our application. We combined the best parts of each of the UI prototypes to build our final user interface.

Color palette



Figure 7: Color wheel and color palette

For the color palette of the application, we selected orange and blue, two complementary colors. As illustrated in figure 7's color wheel, orange and blue are on the opposite sides of the color wheel, providing a high impact due to their contrasting nature. Hence, using these colors would make the elements in our application more prominent and clearer. Orange signifies success, creativity, determination, balance and encouragement, all of which speak to the core values of our application. On the other hand, blue symbolises stability, inspiration, wisdom and provides a calming effect. These also support the objective of our application - to inspire, motivate and achieve success by conquering goals.

4.2. Final Design (with block diagrams & UI)

4.2.1. ER & Class Diagrams

In our application, Tasks (named as “category” in UI) are created by users to set a general goal, before adding Todos using intelligent recommendation or manual input, to set small milestones towards the goal. The app supports basic CRUD (Create, Read, Update and Delete) operations of Tasks and Todos among multiple user devices using Firebase. Besides, time bound features are implemented for each Task and Todo with a deadline and a completion flag. The users are able to set deadlines for each Todo (the day of creation by default) to track the progress. For a Task, the deadline is the earliest deadline of its Todos, to give users a holistic view on the main page of Tasks. Completion flag is set for Todo when it is set by the user, and set for Task once all the Todos have been completed. The progress of a Task is calculated based on the completion ratio of the Todos.

The basic ER diagram for entities visible to the user is shown below. Users can have multiple tasks, and tasks include multiple todos.

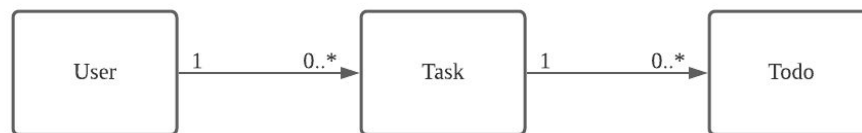


Diagram 1: ER Diagram

The actual class diagram below illustrates the relationship between the entities used in the app. AuthService and DatabaseService classes manage the interfacing to the Firebase database. Guser, Task, and Todo are local data structuring classes. TodoListModel manages the relationship between User, Task, and Todo classes locally, and interfaces with DatabaseService.

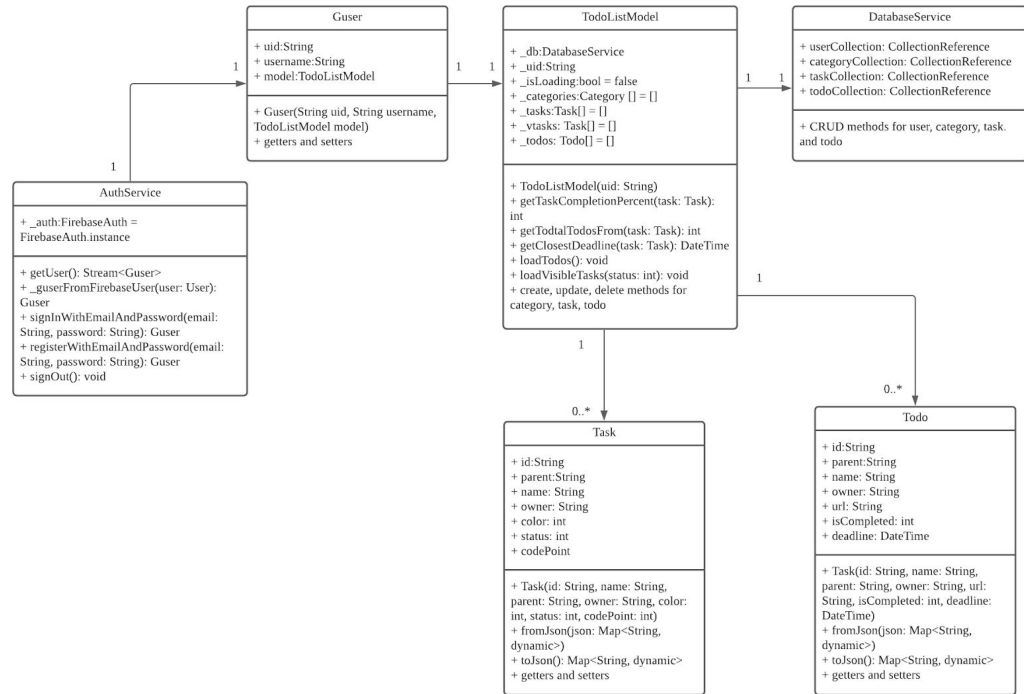


Diagram 2: Class diagram

4.2.2. Site Map & Page Design

The application pages are structured as shown in the following diagram.

Brief explanation of the diagram is as follows:

- Yellow blocks are pages visible to the user.
- Orange blocks encapsulate and manage navigation between yellow blocks.
- The application listens to the status update from Firebase Authentication service, and “Wrapper” navigates between orange blocks depending on the user authentication status.
- “MyApp” is the entry point to the application.

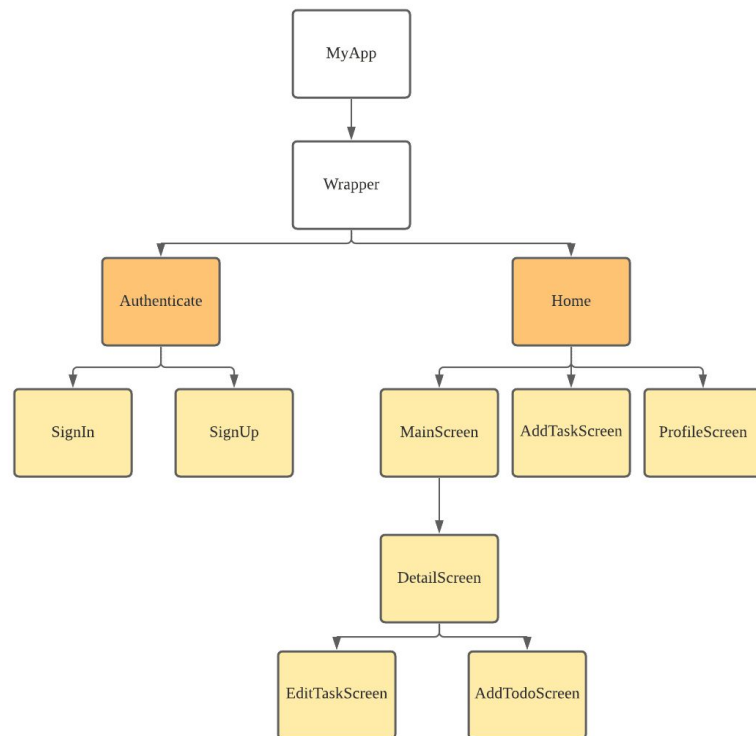


Diagram 3: Application pages structure

Welcome Page (Authenticate)

Welcome page prompts users to input their login or register information respectively in order to use our application and record their stored tasks.

Welcome

Sign in to continue!

Log In

Don't have an Account ? [Sign Up](#)

Welcome

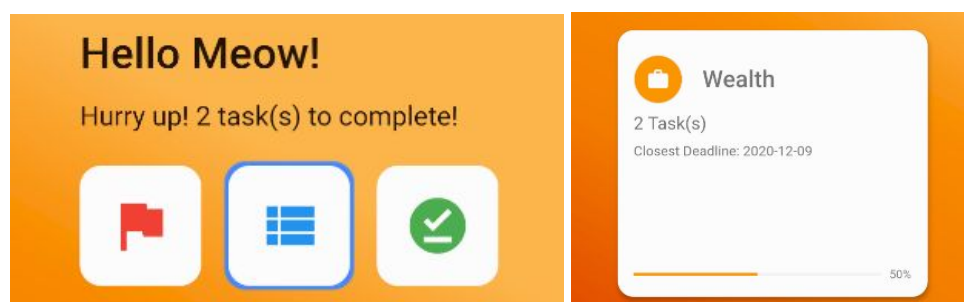
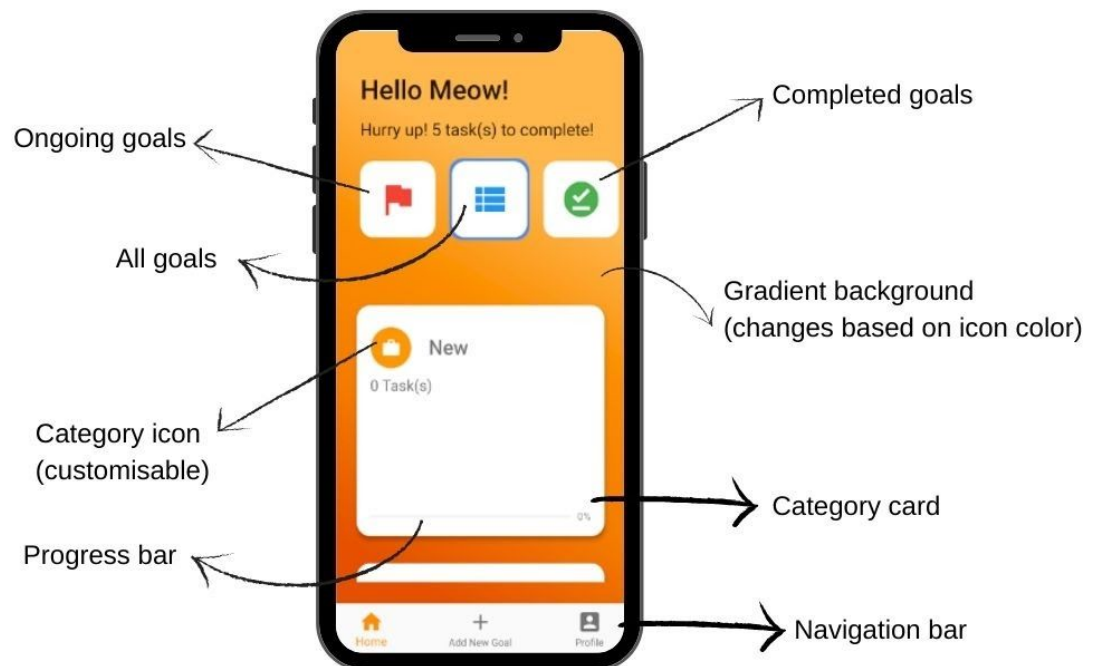
Sign up to continue!

SIGN UP

Already have an Account ? [Sign In](#)

Home Page

After successfully logging in, the user will enter the home page, which consists of a welcome message, filtering buttons and category cards chronologically ordered by their deadlines. The gradient background of the homepage will change based on the selected category (user selects the icon and color while adding a new category).

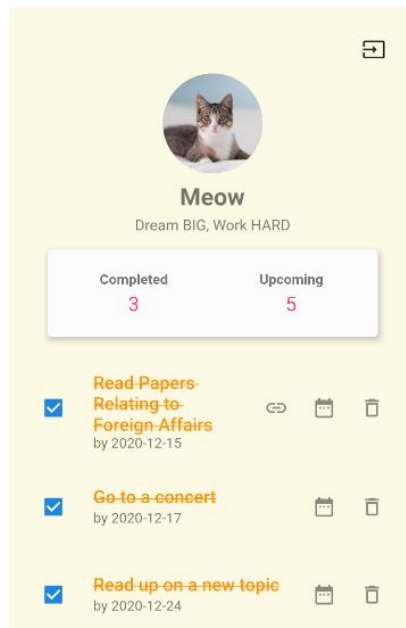


Every page has a bottom navigation bar, which directs the user to the corresponding page upon clicking: Home page, Add New Goal page and Profile page.



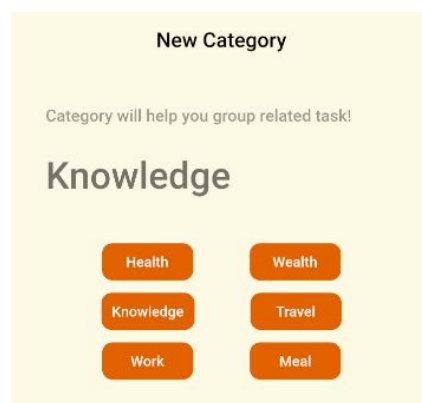
Profile Page

Profile page consists of the user's basic information (including avatar, username, bio), number of completed and upcoming tasks, as well as a list of all the tasks that the user has created.



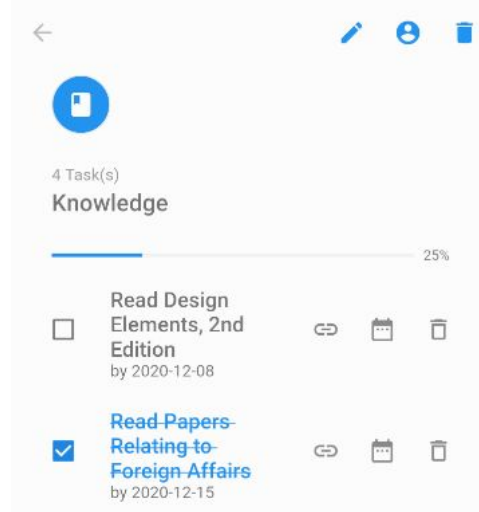
New Category Page

In this page, six main categories will be listed as below left. The user can also choose other categories of their own.



Category Detail Page

Within each category, the user can set different tasks, which are listed in order in the category detail page. In this page, the user can choose their preferred color and icon style as well as modify, add or delete the tasks inside. Notice that for the knowledge category, a link icon will be shown on the corresponding coursera-related tasks, which will direct the user to the specific course on the coursera website.



New Task Page

Users can add new tasks in the specified category. For special tasks, i.e. knowledge, meal, travel, etc, the app will generate suggestions automatically based on the user's input by using API. They could also import Todos via an image stored in the phone, with text recognized and input in the textfield. For users having a demand of time bound features, they can either choose a deadline to finish the task when creating the task or modify later on.

←

New Task

What task are you planning to perform?


Read up on a new topic

Look up for youtube tutorials

Read up on a new topic

Make a summary of relevant notes

Pick a date to finish it



+ Create Task

Final application UI

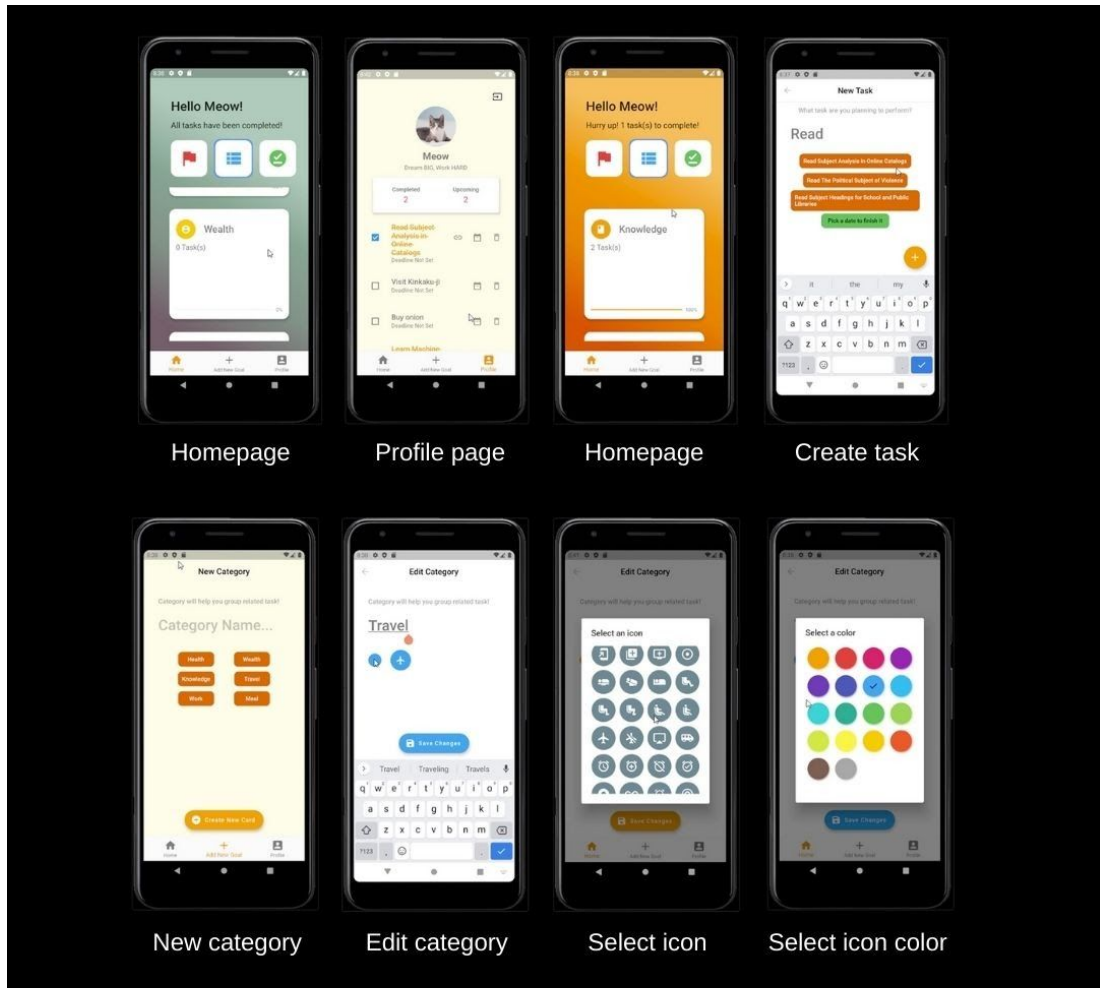


Figure 8: gogoGoals User interface

4.3. Implementation

4.3.1. Implementation of Key Features

Task Recommendation

In order to strengthen the user's spirit of setting a new goal and challenging themselves, tasks are recommended upon user input. The first verb is used as a class identifier, and the rest is used as a search reference for the respective class's API request. For example, if the first input is "read java", then the first verb is "read", so the class is set as "book". The recommendation request is sent to google books API with reference "java" and results can be seen from picture below.

What task are you planning to perform?

read java

Read On the Subject of "Java"

Read Java in a Nutshell

Read Inside Java 2 Platform Security

Pick a date to finish it

Recommendations from external API are limited to 4 classes at the moment, and it can be expanded in the future. Implemented 4 classes are:

- 1.Course (Coursera course search API)
- 2.Book (Google books API)
- 3.Recipe (Spoonacular recipe API)
- 4.Travel (Tripoto API)

An example of code implementation for recommendation feature is shown below:

```
if (cat == "course") {
    response = await http.get(
        'https://api.coursera.org/api/courses.v1?q=search&query=' +
        task.toLowerCase());
} else if (cat == "book") {
    response = await http.get(
        'https://www.googleapis.com/books/v1/volumes?q=subject=' +
        task.toLowerCase());
} else if (cat == "recipe") {
    response = await http.get(
        'https://api.spoonacular.com/recipes/complexSearch?apiKey=8a368b785ac348199b09d5a:
        task.toLowerCase());
} else if (cat == "travel") {
    response = await http.get(
        'https://www.tripoto.com/api/20200803/poi.json?account=LZR64QHZ&token=sqh2dctwmo6r
        task.substring(0, 1).toUpperCase() +
        task.substring(1).toLowerCase());
}
```

```

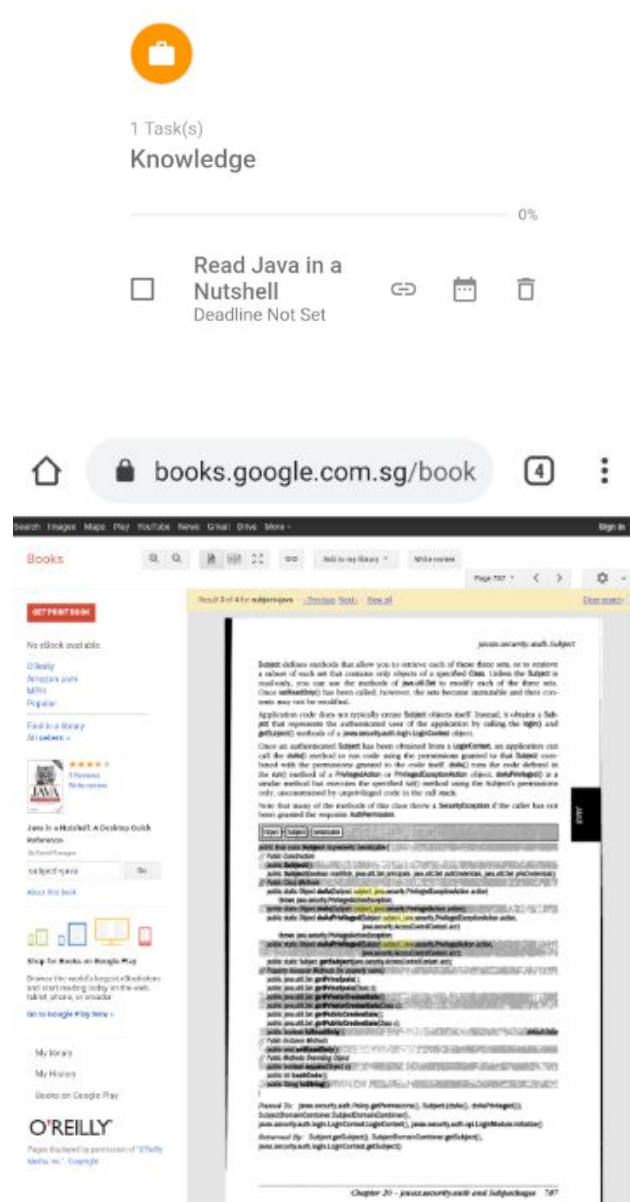
class Course {
    final String cat;
    final String content;

    Course({this.cat, this.content});

    factory Course.fromJson(Map<String, dynamic> json) {
        return Course(
            cat: "https://www.coursera.org/learn/" + json['slug'],
            content: "Learn " + json['name'],
        );
    }
}

```

For course and book, we also included external reference links for the todos. For example, when the user adds a “read Java in a nutshell”, we will add the link to the “Java in a nutshell” pdf next to the todo, so the user can click on the icon and navigate to the book.



The code implementation is shown below:

```

    } else {
      model.addTo(Todo(
        newTask,
        url: url,
        parent: _task.id,
        deadline: deadline,
      )); // Todo
      Navigator.pop(context);
    }
  },
); // FloatingActionButton.extended

value: todo.isCompleted == 1 ? true : false), //
trailing: Wrap(
  children: <Widget>[]
    todo.url != "" && todo.url != null
      ? IconButton(
        icon: Icon(Icons.link),
        onPressed: () {
          launchURL(String url) async {
            if (await canLaunch(url)) {
              await launch(
                url,
                // forceWebView: true,
              );
            } else {
              print('Could not launch $url');
            }
          }
        },
        print(todo.url);
        launchURL(todo.url);
      ),
    ) // IconButton

```

For the categories that do not have API resources, a recommendation api in node.js is used to find relevant recommendations from our recommendation database. We also implemented some keyword matching so for those categories without external apis, we can also recommend some relevant todos based on the user input. We also implemented a random algorithm so for the same user input, we will provide random recommendations for the user. The code implementation is shown below:

```

app.get("/cat/:keyword", (req, res) => {
  (async () => {
    try {
      var keyword = req.params.keyword.toLowerCase();
      if (keyword.includes("health")) {
        keyword = "health";
      } else if (keyword.includes("knowledge") || keyword.includes("study")) {
        keyword = "study";
      } else if (keyword.includes("personal")) {
        keyword = "personal";
      }
      else {
        keyword = "random";
      }
      let query = db.collection("recommendation").where("cat", "==", keyword);
      let response = [];
      let result = [];
      const snapshot = await query.get();
      // if (snapshot.empty) {
      //   query = db.collection("recommendation").where("cat", "==", "random");
      // }

      await query.get().then((querySnapshot) => {
        let docs = querySnapshot.docs;
        for (let doc of docs) {
          const selectedItem = {
            cat: doc.data().cat,
            content: doc.data().content,
          };
          response.push(selectedItem);
        }
        return null;
      });
      get_random = function (list) {
        return list[Math.floor(Math.random() * list.length)];
      };

      for (i = 0; i < 3; i++) {
        var temp = get_random(response);
        result.push(temp);
        response = response.filter(function (e) {
          return e !== temp;
        });
      }

      return res.status(200).send(result);
    }
  })();
});

```

Synonym Matching

For better user experience and more intelligent recommendation, the recommendation can be done with the synonyms of the keyword.

Upon user input, an API request is sent to retrieve synonyms of the input. If synonyms include one of the mappable keywords, recommendations are made based on it.

API used: <https://words.bighugelabs.com/>

What task are you planning to perform?

study

Learn Machine Learning

Learn Indigenous Canada

Learn The Science of Well-Being

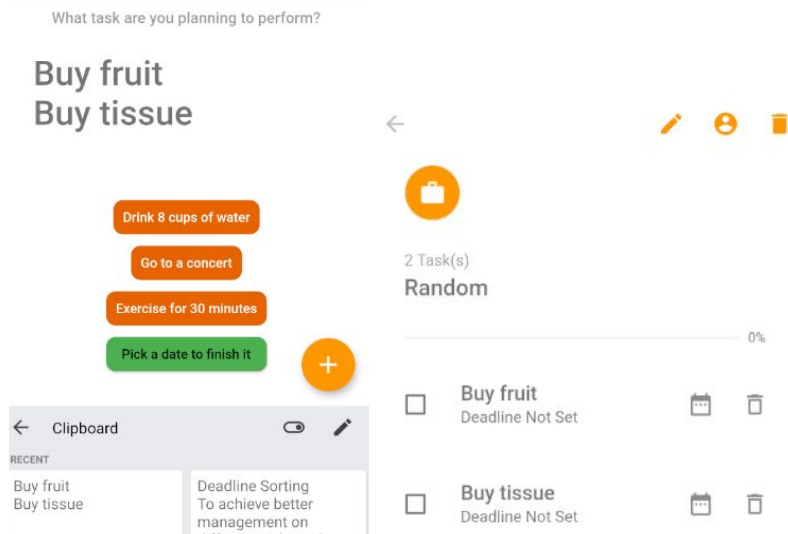
Pick a date to finish it

Part of code implementation for synonym feature is shown below:

```
if (cat != "course" && cat != "book" && cat != "recipe" && cat != "travel") {
  var apicheck = await http.get(
    'https://words.bighugelabs.com/api/2/75d5165665b1923827701a0e530d6ca6/' +
      cat.toLowerCase() +
      "/json?");
  // print(apicheck.body);
  Map<String, dynamic> result = jsonDecode(apicheck.body);
  String checklist = result["verb"]["syn"].toString();
  print(checklist);
  if (checklist.contains("learn")) {
    cat = "course";
  } else if (checklist.contains("read")) {
    cat = "book";
  } else if (checklist.contains("cook")) {
    cat = "recipe";
  } else if (checklist.contains("travel") || checklist.contains("visit")) {
    cat = "travel";
  }
}
```

Add multiple todos in one-go

When a user copies multiple-line text at once by pasting it into the text input field or typing multiple-line text, they are able to create the texts as a multiple Todo list. This would speed up creating multiple todo lists from other sources.



```
TextField(
  keyboardType: TextInputType.multiline,
  controller: myController,
  minLines: 1,
  maxLines: null,
```

```
// _scaffoldKey.currentState.showSnackBar(snack
} else {
  LineSplitter ls = new LineSplitter();
  List<String> tasks = ls.convert(newTask);
  for (var i = 0; i < tasks.length; i++) {
    if (tasks[i] != "") {
      model.addToDo(Todo(
        tasks[i],
        url: url,
        parent: _task.id,
        deadline: deadline,
      )); // Todo
    }
  }
}

Navigator.pop(context);
```

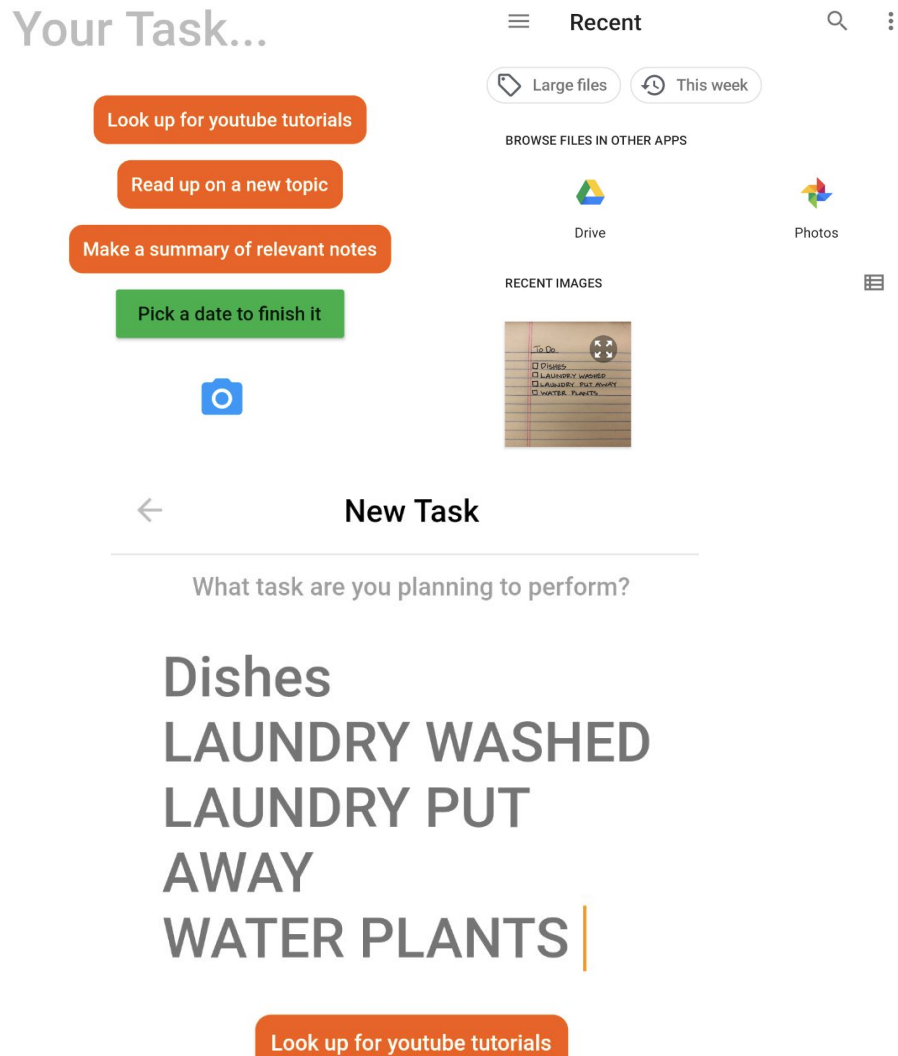
Handwritten Note Conversion

For the convenience of users who write to-do lists on paper notes, text recognition transfers notes from images stored in the phone to multiple lines multiple-line text.

Upon user input, an API request is sent to retrieve recognized text of the input image. The recognised text is automatically pasted into the text input field, allowing the user to create a

multiple Todo list or modify based on computer vision result. This would speed up creating multiple todo lists from other sources.

API used: <https://developers.google.com/ml-kit/vision/text-recognition>



```

class _AddIodoScreenState extends State<AddIodoScreen> {
  io.File pickedImage;
  var text = '';

  bool imageLoaded = false;

  Future pickImage() async {
    var awaitImage = await ImagePicker.pickImage(source: ImageSource.gallery);

    setState(() {
      pickedImage = awaitImage;
      imageLoaded = true;
    });
    FirebaseVisionImage visionImage = FirebaseVisionImage.fromFile(pickedImage);
    TextRecognizer textRecognizer = FirebaseVision.instance.textRecognizer();
    VisionText visionText = await textRecognizer.processImage(visionImage);

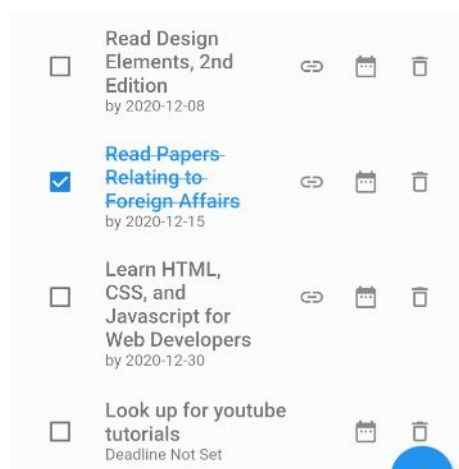
    for (TextBlock block in visionText.blocks) {
      for (TextLine line in block.lines) {
        for (TextElement word in line.elements) {
          setState(() {
            text = text + word.text + ' ';
          });
        }
        text = text + '\n';
      }
    }
    textRecognizer.close();

    print(text);
    myController.text = text;
    text = '';
  }
}

```

Deadline Sorting

To achieve better management on different tasks with different deadlines, all the tasks are sorted based on their deadlines while all the categories are sorted based on their closest deadline (i.e. the earliest deadline of its tasks).



The tasks with no specified deadline will give lower priority compared to those with deadlines set. The sorting algorithm is achieved as shown below.

```

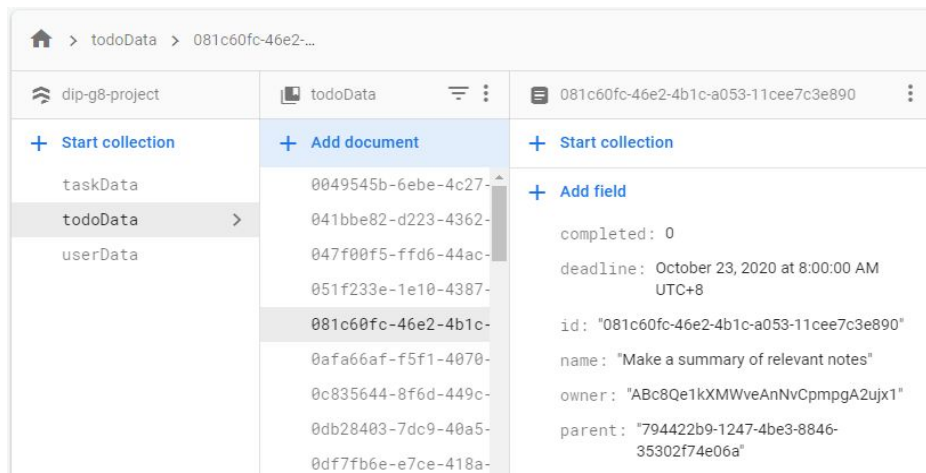
var _todos =
  model.todos.where((it) => it.parent == widget.taskId).toList();
_todos.sort((a, b) {
  if (a.deadline == null && b.deadline == null)
    return 0;
  else if (b.deadline == null)
    return -1;
  else if (a.deadline == null) return 1;
  return a.deadline.compareTo(b.deadline);
});

```

4.3.2. Implementation of Database

User Information Data

Database is designed to resemble a relational database and stored in Cloud Firestore. Each type of data (user, task, todo) has a collection, and each collection has entries as documents. Entries have the essential information such as id and name, also the relational information such as parent (the todo's parent task) or owner (the task owner's user ID).



User login information such as email and password are not stored together with optional information on Cloud Firestore. They are managed by Firebase's own Authentication service.

Identifier ↓	Providers	Created	Signed In	User UID
test@test.test	✉	Sep 30, 2...	Dec 6, 2...	muaGPV6fV7OJ7ZHtqN4...
test@test.com	✉	Dec 5, 2...	Dec 5, 2...	VFsxHgT826YvSnORP5y...
cly@gmail.com	✉	Sep 29, 2...	Sep 29, 2...	5uYLvyTiumR09tkhT3a0Y...

Recommendation Data

For the categories that do not have API resources, a recommendation database is provided in order for our recommendation api to give relative suggestions. Each data contains two attributes: category and content. When related categories are chosen, three suggestions will be prompted to the screen for users to pick.

Collection	Document	Field
+ Start collection	+ Add document	+ Add field
recommendation >	0	cat: "health"
	165kPIF4berAoaWH6Rrn >	content: "Consume less salt and sugar"
	1JQx3TsHZH4z6d4xCuI6	
	1XusW0n6y5WUu1juMDks	
	2IpJUh6CYHXR2SFVwwKz	
	3g0vIbTHjyMe9txDwmbb	
	6uCB7M4zxXvPIH0HKFaI	
	9FLhJlPjN8mGHI6t3p5	

4.4. Discussion

Paired with a user friendly user interface (UI), the application's overall goal was to help users get rid of their procrastination by effectively tracking their tasks, goals and completions. This was achieved by the end of the semester with the completion of the app, along with additional features such as Task Recommendations with Synonym Matching, and Handwritten Note Conversion enabled. Overall, our application functions are released as expected, with minimal

issues regarding backend logic and display issues. However, the app can be improved in many areas. This includes the Backend, Frontend and UI.

Backend

In the backend, the app uses loops, logic, classes, functions and APIs to compare, search and make decisions on the priority of tasks, type of tasks, task suggestions and word recognition based on the user's input. Currently, the app's backend logic has proven to be effective and robust in its tasks. The app has also applied emerging technology, Artificial Intelligence (Computer Vision), in a useful and ethical manner. This machine learning algorithm allowed users to digitize their conventions easily with a satisfactory conversion accuracy. However, improvements can be made in the extensiveness of the APIs' to search for more suggestions in a broader range of categories, as well as using Artificial Intelligence recommender systems to detect patterns in the user's behaviour to recommend better tasks for the user based on their personal preferences. In such a case, the user data privacy will become an unignorable concern when it comes to exchanging user data on database servers.

Frontend, UI

The app has a simple user interface which allows users to easily discern how the app is to be used without any prior instructions given. It also provides a bright orange theme (by default, which can be customised) in the app's background and a complementary colour for its task and category selection, which motivates the user to write or complete a task when they look at it. However, the style of the buttons and the layout of the app could be better improved to be more visually appealing to the user. Other improvements include allowing the users to be able to choose and change the coloured theme for the app, instead of keeping its current default colour, creating better designs for the icons in the app to prevent confusion or unfamiliarity when the user is unsure of the purpose of a certain button, and including voice recognition to allow for a wider range of users such as the disabled or visually impaired.

Apart from these, the app's Synonym Matching and Task Recommendations could also be improved as it is restricted to the user's input and the suggestions suggested are fixed to the top three results found in the API. For example, if a user was to search up "Visit Japan" the top

three recommendations would not differ when the same input is entered again. An improvement to this would be to include Artificial Intelligence, specifically, Natural Language Processing. With this, the recommendations would be more diverse and the app would be able to learn from the user inputs.

As for Task Recommendations, an improvement would be to include more APIs for the other Categories. However, considering that we have to use a specific type of API in websites or databases that is able to have links and titles that will be able to link up with the app, the APIs would not be easy to look for, thus hindering the further development of the Task Recommendations feature.

5. Design and Implementation

5.1. Conclusion

The main objective of the mobile application focuses on increasing personal productivity of a user and our sole goal was to allow others to achieve their goals. The mobile application that we have created contains various useful features such as API Integration, Task Recommendation, Synonym matching and Handwritten Note Conversion that will help many individuals in their quest to better themselves and further improve the efficiency of the application. Individuals would be able to improve themselves in areas such as Health, Knowledge, Travel and more. Not only would our app allow users to keep track of their goals, it would also allow users to be more productive and efficient at the same time thanks to our intuitive design. All in all, gogoGoals is here and is ready to help you go for your goals.

5.2. Recommendation for Future Works

We have tried to implement many features that distinguish our app from other goal trackers, yet there are still limitless possibilities for the future of our app. An extensive API database for all possible categories of tasks can be put in place, and the emerging Natural Language Processing technologies at a higher level of intelligence can also be implemented to provide users with personalised recommendations based on individual behaviour. Our app can also provide voice recognition and translation services which allow for enhanced ease of use, benefitting all age groups as well as persons with disabilities.

Apart from all these, there can also be implementations of mini games, where users get motivated by in-game rewards for completing their tasks. Also, there may be collaborations possible with other developers or softwares, which can allow users to complete their tasks from the app itself, where they do not have to maneuver between apps, enhancing the usability of our app. All in all, the future of our app knows no bounds, and through further implementations of features, our app will be able to cater to the majority needs of different types of users.

References

Erik.P, Richard.H, I-Mei.L & Padma.D (2014) Increase Productivity, Decrease Procrastination, and Increase Energy. Biofeedback: Summer 2014, Vol. 42, No. 2, pp. 82-87

MindTools (n.d.), Personal Goal Setting | Planning to Live Your Life Your Way. Retrieved from <https://www.mindtools.com/page6.html>

ScienceDaily (2015, October 29), Frequently monitoring progress toward goals increases chance of success. Retrieved from <https://www.sciencedaily.com/releases/2015/10/151029101349.htm>

Kelly B (Sep 1, 2000), The Effects of an Academic Procrastination Treatment on Student Procrastination and Subjective Well-Being, Carleton University, Ottawa, Ontario.

Flutter

<https://www.freecodecamp.org/news/what-is-flutter-and-why-you-should-learn-it-in-2020/>

Strides <https://www.stridesapp.com/>

Microsoft To Do <https://www.fool.com/the-blueprint/microsoft-to-do-review/>

Habitica <https://habitica.com/static/home>

Appendices:

A. Bill of Materials

This item is not applicable to the project.

B. Design Diagrams

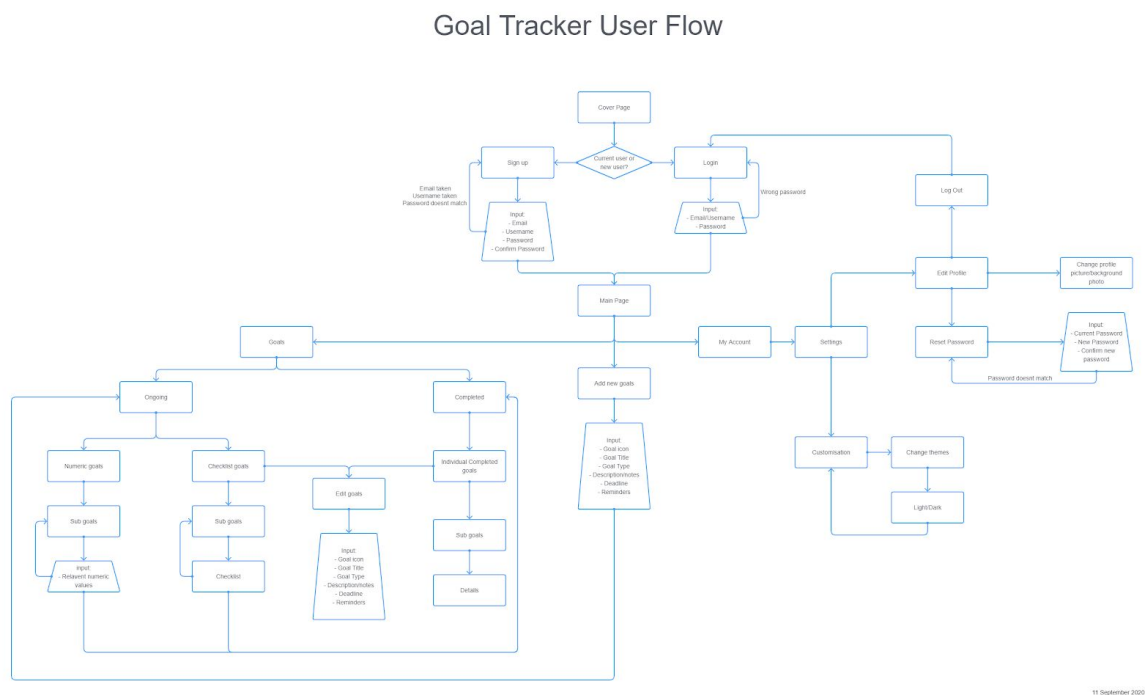


Figure: Sitemap

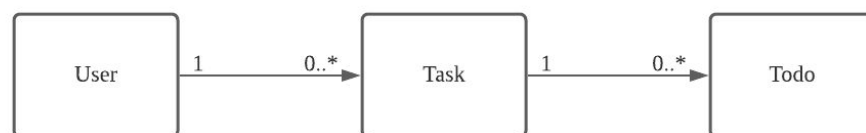


Figure: ER diagram

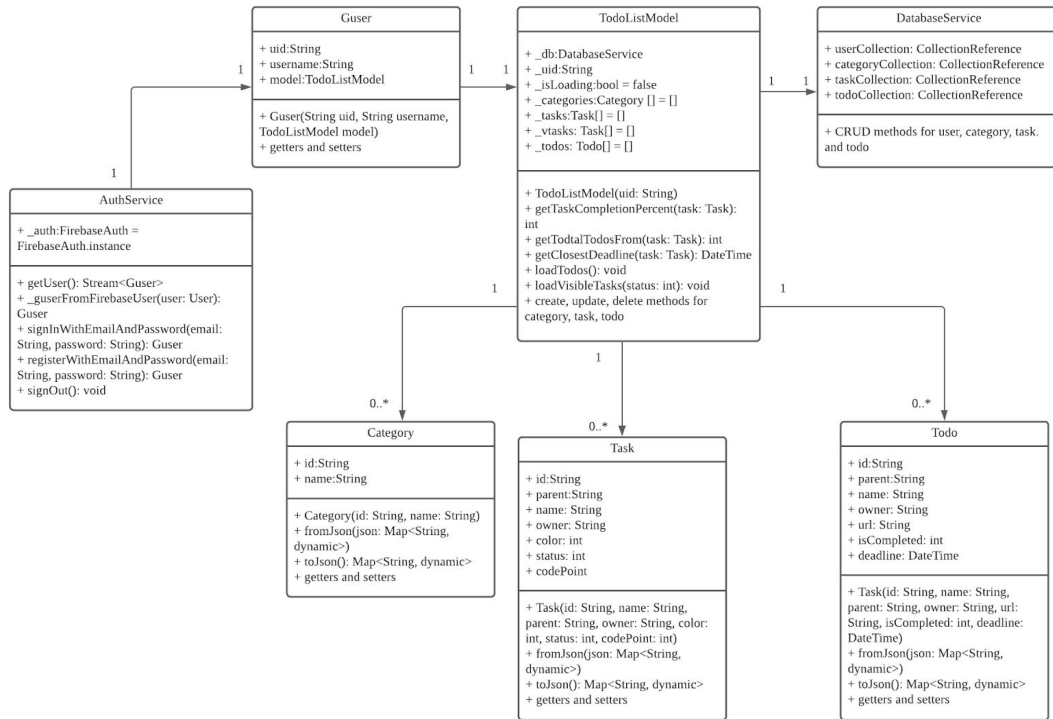


Figure: UML class diagram

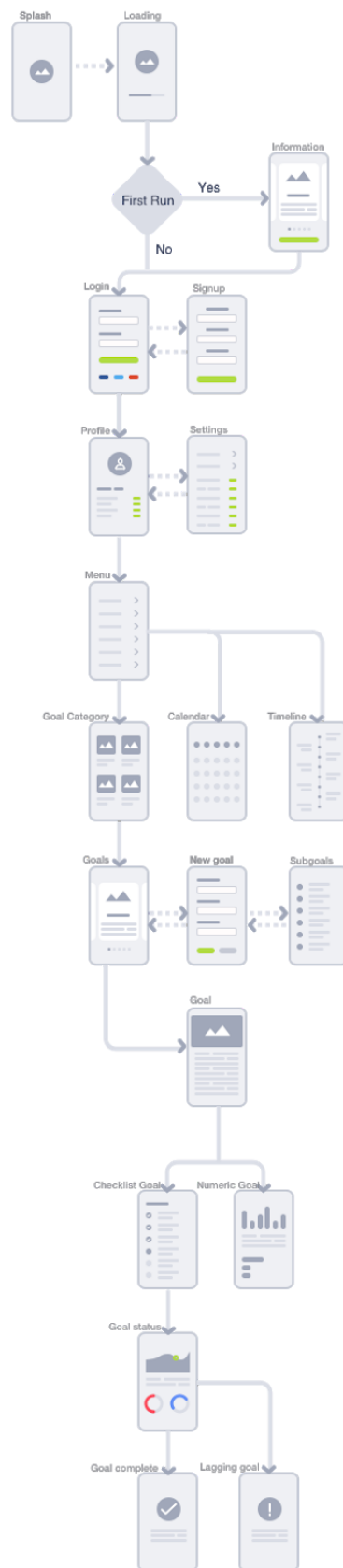


Figure: App wireframe

C. User Guide

gogoGoals is a Design and Innovation Project for crafting beautiful, intelligent user experiences for tracking goals and manage personal productivity. We aim to develop a flexible, fast, and modularized mobile application that can allow users to easily manage and view goals by category, with intelligent recommendation of to-dos. gogoGoals works with Android and iOS, and is free and open source.



Getting Started

Prerequisites

Android

API level 16 (Jelly Bean) or later, with Google Play services

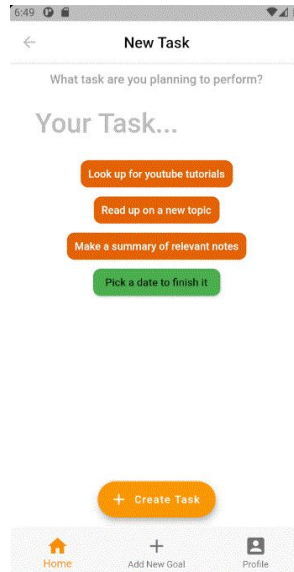
iOS

iOS 10 or later

Features

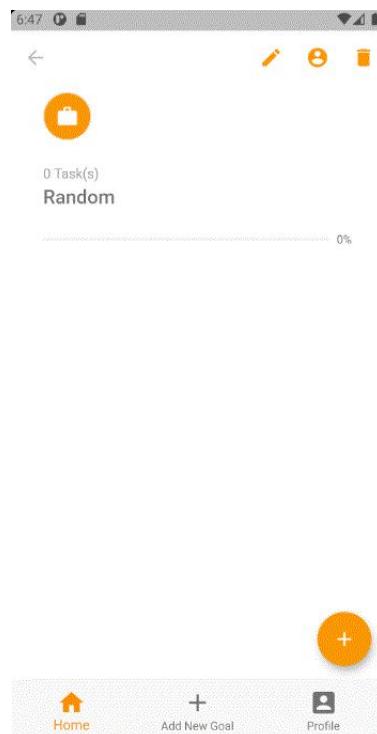
Intelligent Recommendation

gogoGoals automatically detects keywords to search for suitable to-dos via API sources, with support of synonym matching. (Try "Learn Flutter programming" or "Study Flutter programming")



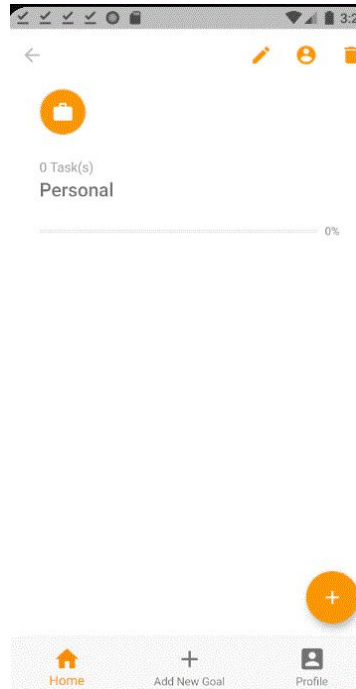
Add Multiple Todos in One Go

gogoGoals allow users to add multiple Todos at one go. You can copy multiple line text from other apps or you can type on your own texts and paste it onto the text input field and it will automatically create multiple Todos.



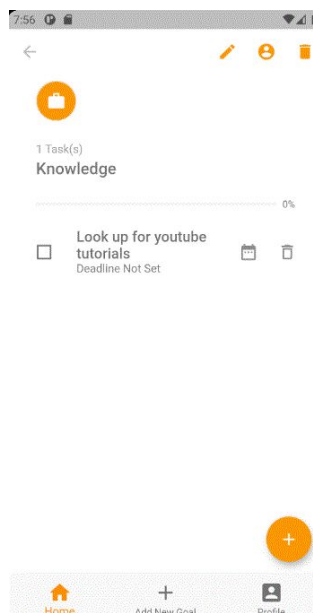
Machine Learning OCR Algorithm for Handwritten Note Conversion

Machine learning algorithms transfer your handwritten Todo notes to gogoGoals fast and handily, with Google ML Kit Vision API.



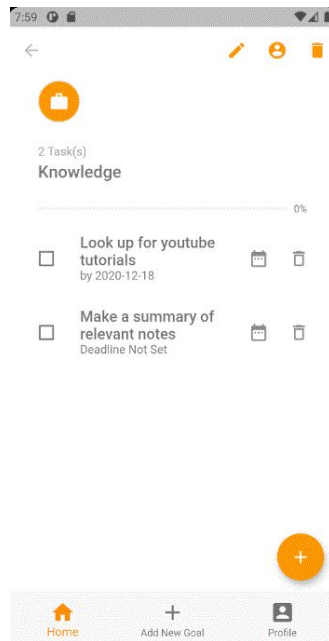
Time Bound

Deadline allows users to keep on time with their commitments. The earliest deadline of each task category is displayed on the main page at a glance. (Try checking Profile page after setting deadlines to your tasks)



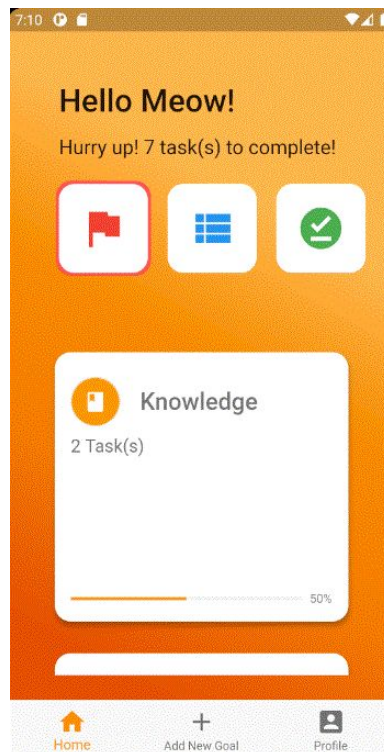
Progress Bar

gogoGoals ensure users to keep track on their progress under both categories and tasks.



Highly Customisable UI

Higher customizability allows you to have a personal touch by changing icons and their colours based on your preferences.



Contribute to gogoGoals

A few resources from Flutter to get you started:

- [*Lab: Write your first Flutter app*](#)
- [*Cookbook: Useful Flutter samples*](#)

For help getting started with Flutter, view our [online documentation](#), which offers tutorials, samples, guidance on mobile development, and a full API reference.

D. Maintenance Guide

If you find any bugs or issues, please open an issue or send an email to us (lchen025@e.ntu.edu.sg) with detailed information. We appreciate your help!

E. How-To Guides

Intelligent Recommendation

gogoGoals automatically detects keywords to search for suitable to-dos via API sources, with support of synonym matching. (Try "Learn Flutter programming" or "Study Flutter programming")

Add Multiple Todos in One Go

gogoGoals allow users to add multiple Todos at one go. You can copy multiple line text from other apps or you can type on your own texts and paste it onto the text input field and it will automatically create multiple Todos.

Machine Learning OCR Algorithm for Handwritten Note Conversion

Machine learning algorithms transfer your handwritten Todo notes to gogoGoals fast and handily, with Google ML Kit Vision API.

Time Bound

Deadline allows users to keep on time with their commitments. The earliest deadline of each task category is displayed on the main page at a glance. (Try checking Profile page after setting deadlines to your tasks)

Progress Bar

gogoGoals ensure users to keep track on their progress under both categories and tasks.

Highly Customisable UI

Higher customizability allows you to have a personal touch by changing icons and their colours based on your preferences.

Contribute to gogoGoals

A few resources from Flutter to get you started:

- [*Lab: Write your first Flutter app*](#)
- [*Cookbook: Useful Flutter samples*](#)

For help getting started with Flutter, view our [online documentation](#), which offers tutorials, samples, guidance on mobile development, and a full API reference.

F. Source Code

Source code is available on the Github repositories below.

Source code for the application: NTU-AY2020-DIP-Group-8/gogogoals, link:

<https://github.com/NTU-AY2020-DIP-Group-8/gogogoals>

Source code for the api: NTU-AY2020-DIP-Group-8/api, link:

<https://github.com/NTU-AY2020-DIP-Group-8/api>

G. Others

Video credits:

HowardFreeman Motivation: <https://www.youtube.com/watch?v=gF0rrpMH-Jo&t=55s>

Rafee Rahman: <https://www.youtube.com/watch?v=lcU3pruVyUw>

Travel Alberta: <https://www.youtube.com/watch?v=mJhFtv5UTk8>

Little Free Library: https://www.youtube.com/watch?v=kul-g_30HuU