

UNIVERSITY POLITEHNICA OF BUCHAREST
FACULTY OF AUTOMATIC CONTROL AND COMPUTERS



University schedule personalization and automatization for productivity

Graduate:

Iuga Bogdan-Andrei

Thesis supervisors:

Lecturer, Dr. Vlad Posea

Engr. Ioana Alexandru

Bucharest

2021

RO: Această lucrare prezintă implementarea și integrarea unui modul dedicat îmbunătățirii modului de gestionare a activităților universitare într-o aplicație deja existentă. Principalele funcționalități implementate au fost adăugarea unor evenimente separate pentru sarcini universitare precum teme și proiecte, posibilitatea de a alege obiective pentru acestea și crearea de elemente vizuale care să ajute utilizatorul să găsească informațiile dorite mai rapid.

Aplicația fiind deja utilizată de către studenții din Facultatea de Automatică și Calculatoare, Universitatea POLITEHNICA din București, au fost analizate structurile administrative din cadrul facultății și alte aplicațiile care au aceleași funcționalități ca și cele propuse în această lucrare, pentru a satisface eficient nevoile studenților. Tema a fost abordată din diverse puncte de vedere, precum cel al arhitecturii utilizate, a interfeței cu utilizatorul, al tehnologiilor. Au fost urmate convenții stabilite de dezvoltare ale aplicației astfel încât rezultatul să funcționeze alături de celelalte module și să poată fi livrat către utilizatori.

ENG: This paper presents the implementation and integration of a module dedicated to improving the way of handling university activities inside an already existing application. The main functionalities implemented were the addition of separated events for university tasks such as homework and projects, the possibility of choosing goals for them, and the creation of visual elements in order to help the user find relevant information faster.

The application being already used by the students of the Faculty of Automatic Control and Computer Science, University POLITEHNICA of Bucharest, there were analyzed the administrative structures inside the faculty and also other applications that have the same functionalities like the ones proposed in this paper, in order to satisfy the needs of the students. The theme was analyzed from different points of view, such as the used architecture. the user interface, the technologies used. The development conventions of the application were followed so the result works alongside the other modules and can be delivered to the users.

Contents

List of figures	5
List of tables	7
1 Introduction	8
1.1 Context	8
1.2 Background	9
1.3 ACS UPB Mobile	10
1.4 Proposed functionality	10
1.5 Goals	11
1.6 Structure	12
2 Scope	13
2.1 Structure	13
2.2 Grading	14
2.3 Tasks, Assignments, and Deadlines	15
2.4 Resources	16
3 State of the art	18
3.1 General productivity applications	19
3.1.1 Target Audience	20
3.1.2 Functionalities	20
3.1.3 UI & UX	22
3.1.4 Conclusion	22
3.2 Specific univeristy applications	22
3.2.1 Functionality	23
3.2.2 User engagement	25

3.2.3 Conclusion	25
3.3 Technology	26
4 Interface	28
4.1 Design conventions and strategies	28
4.2 Timetable	30
4.3 Red Routes	31
4.4 Wireframes	32
4.5 Final UI	35
5 Implementation	37
5.1 ACS UPB Mobile architecture	37
5.1.1 Database	37
5.1.2 Provider	38
5.1.3 BLoC Design Pattern	39
5.1.4 Flow	40
5.2 Integration strategy	40
5.3 Database integration	41
5.4 System integration	44
6 Contribution process	47
6.1 Repository	47
6.2 Pull Request	49
6.3 Deployment	49
7 Conclusion	51
7.1 Achievements	51
7.2 Prospect	53
7.2.1 Additional features	53
7.2.2 StundentHub	54
Bibliography	55
Appendices	58
A Calendar interfaces	59
B UML diagrams	61

List of Figures

3.1	Home page	23
3.2	Timetable page	23
3.3	Filter page	23
3.4	Event details page	24
3.5	Class details page	24
3.6	Active users	25
4.1	Screenshot of the timetable	30
4.2	Red Routes matrix	31
4.3	Home page	32
4.4	Planner page	32
4.5	Event Details page	33
4.6	Class Information page	33
4.7	Add/Edit event page	34
4.8	Add/Edit goal	34
4.9	Homepage	35
4.10	Planner	35
4.11	Timetable	35
4.12	Event page	36
4.13	Class page	36
4.14	Add/Edit event	36
4.15	Add/Edit goal	36
5.1	Screenshot of the events implementation	39
6.1	Planner project	48
A.1	Google Calendar interface	59

A.2	Apple Calendar interface	60
B.1	Class diagram	61
B.2	Use-case diagram	62

List of Tables

3.1	Calendar relevant functionalities	21
5.1	Task event structure	42
5.2	Users new field	43
5.3	Goals structure	43
5.4	taskGoals structure	44

Chapter 1

Introduction

1.1 Context

In the opinion of Peter Turla [1], *You don't manage time, you drive behavior*. For the purpose of this paper, we can consider the previous observation as the general idea behind our proposed solution of increasing productivity by using customized time management tools.

As technological advances have become an integrated and indispensable part of our lives [2], a new problem arises, that is, whether we should adopt a new technology or use an old one. Therefore, we need to consider all the factors that come into play when making a transition. Is the new solution adding enough value? Is it mature enough? Do I have enough time to accommodate myself with it? Will it disrupt other processes that I use? Will another one take its place too soon? Is it reversible? Such questions come to mind when making a decision that can have an overwhelming impact [3].

Taking a look at the traditional way of handling tasks, there is no standard. However, it can be summarised as the battle between procrastination and productivity [4]. Putting it into a student's perspective, many factors contribute to optimizing the time used for learning, and using technology to improve the education process is one effective way. Translating natural flows of thoughts and actions regarding scheduling into digital workflows unlocks new capabilities. It also comes into aid for those that are having a hard managing time by themselves.

This paper proposes an upgrade to an already existing cross-platform application designed to englobe university resources. It is meant to give the students a personalized and user-friendly way of scheduling and automating university assignments and classes.

1.2 Background

As every software solution that aims to help other people starts, the first and most important milestone is deciding on the development tools and processes. However, given the plethora of technologies available to every developer, choosing one gets harder and harder as many questions vital to the relevance and success of the final product start arising [5].

First, and most importantly, should this module be a stand-alone application or part of a more extensive suite of functionalities packed together inside a single, core application? Looking at the most popular apps used by young adults [6], we can see the trend of encapsulating as much functionality as possible but still keeping these functionalities independent, so the users are not overwhelmed by long and tedious flows caused by the modules of the application being too strongly coupled or poorly designed. Seeing this trend and understanding the utility of having many functionalities related to one field, education in this case, even though this module could have been, on its own, an application, this would defeat its purpose of helping the students be more productive. Furthermore, having too many applications focused on solving different sections of one general problem gives an overhead to the user, switching between platforms with different interfaces constantly.

Secondly, on what framework should it be developed to be competitive with today's standards, easily maintainable, adjustable, and most importantly, accessible to the consumer. Looking at the smartphone market in Romania, over 97% of the population has at least one smart portable gadget [7].

Taking a closer look, we see that two operating systems dominate, making together 99.18% of Mobile operating systems' market share worldwide in the first quarter of 2021 [8]. Android, present in 72.72% of all smartphones, and iOS in

26.46%, have a clear duopoly in this sector, so the natural choice is to deliver the solution proposed in this paper on a mobile platform, ensuring that users from both environments receive the same experience and quality.

1.3 ACS UPB Mobile

Started also as a Bachelor's degree concept to help students gather different resources into one place, to make them more easily accessible, ACS UPB Mobile¹, referred to in this paper as the application or the app, is now published as a free tool dedicated to the students of the Faculty of Automatic Control and Computer Science² of University POLITEHNICA of Bucharest³.

We were inspired by the approach and development done by the team behind ACS UPB Mobile, and decided that the best strategy for delivering our solution is to integrate it inside the application. A collaboration was proposed, and as the goal of this paper is ultimately to help students with their academic activities, which is the same as the application's purpose, we reached an agreement.

The project owner agreed to guide and supervise the development process, and decided that a feature can be delivered only if it passes all the quality checks. The complete development process, starting from planning to delivery, will be fully described in the 6'th chapter.

1.4 Proposed functionality

This paper documents the development of a schedule management module, putting emphasis on the following list of goals and functionalities that seek to increase the productivity of its users:

- Integrating assignments with the already existing events types, making sure users can add, edit, see and delete them.
- Adding a planner functionality, letting the user choose what assignments they want to focus on and what to ignore.

¹<https://github.com/student-hub/acs-upb-mobile>

²<https://acs.pub.ro/>

³<https://upb.ro/en/>

- Adding customization to the assignments, specific to each user, letting them set their own goals when solving a task.
- Giving relevant statistics to the users for each assignment, such as estimated effort required, pace, and advices.
- Adding relevant pieces of information in key locations inside the application.
- The mentioned features are coherently integrated, using the code, design, and resource conventions already existing in the project, making the application as homogeneous as possible.

1.5 Goals

As time passes and new generations adhere to higher education degrees, eager to learn and explore the full potential of technological advancements, any university will struggle to maintain any informational application daily updated, assuring that the distributed information is correct and up to date. Even with a team dedicated only to the application's administration, this type of architecture is not vertically efficient, as it is prone to misunderstandings and errors.

This is why giving students accountability is, from the other similar applications point of view, such an innovating factor. Thoroughly described in the following chapters, the app filled a gap in the distribution of academic resources. As an open-source project, student-centered, modularized, cross-platform, with the alpha version already used by the students, it will be the working base of this paper. We will be laying the foundation of the schedule personalization and automatization module that is to be integrated within it.

Our main goal is to integrate the proposed functionalities described in the previous section while also preserving the application's architecture. We want our solution to naturally blend with the other components while also engaging the users to experiment and take full advantage of it. Other goals revolve around general interface changes to improve the user experience.

1.6 Structure

This paper will be structured in 7 chapters, as follows:

- Chapter 1: Introduction
- Chapter 2: Scope, where we will be analyzing the academic establishment to better understand the needs of the students.
- Chapter 3: State of the art, where we will take a closer look at the current popular similar application that provides a more generalized solution for what is of concern in this paper.
- Chapter 4: Interface, where we will explain how we consolidated our upgrade inside the application and the UI & UX guidelines we followed.
- Chapter 5: Implementation, where we will present the implementation and integration of our solution.
- Chapter 6: Deployment, where we describe the processes of delivering our features to the users.
- Chapter 7: Conclusions, where we present what we achieved in this paper and what opportunities it further enables.

Chapter 2

Scope

Based upon the work done in “Design and Implementation of a Cross-Platform Mobile Application That Facilitates Student Collaboration”, the bachelor thesis of Ioana Alexandru [9], this paper will continue analyzing the development process, prototyping, and impact of a personalization and automation feature within a computer science engineering faculty in Romania, namely Faculty of Automatic Control and Computer Science (the faculty or ACS) of University POLITEHNICA of Bucharest (UPB).

To define the scope of the software solution implemented in this paper, we first need to establish the needs of the direct beneficiaries, namely students. We do that by analyzing their environment, starting from an organizational point of view.

2.1 Structure

Romania is part of the European Higher Education Area¹ and UPB being compliant with the Bologna Process [10], the faculty offers three different learning cycles, namely a Bachelor’s degree, a Master’s degree, and a Doctoral school. As main points of interest for this paper, we will look at the first two, who are structured as follows :

¹<https://www.ehea.info/>

- Bachelor's degree, a four-year cycle, split into two domains² :
 - Systems Engineering (IS)
 - Computers and Information Technology (CTI).

From an organizational perspective, each generation of students from both branches is split into large assemblies, up to 150 people, referred to as a series. Each series is fragmented into smaller collectives, containing around 30 persons, called groups, which can be split into even a more minor organizing form known as a semi-group. There is one student responsible for the representation in front of the professors and disseminating information towards the colleagues for every one of them.

- Master's degree, only two years long, and it comes as a natural continuation to the final of the Bachelor's program, being organized into Masters of Science programs, each one following a specific curriculum. It keeps the same structure presented above, but scaled down, as the students are more widely distributed.

2.2 Grading

A university year for an ACS student is structured in two semesters, each succeeded by three weeks dedicated to exams, called exam sessions. Each semester has up to five main classes and other optional and facultative subjects. To consider a class passed, the student has to meet the criteria established by the course's coordinator. These criteria revolve around gathering a minimum of points throughout the semester and at the final exam.

The most common graded activities are laboratories and seminars, doing independent work in the form of assignments such as homework, projects, and research, but can also they can be evaluations such as tests and midterms. At the end of the semester, the grades for all of these activities are summed together, and the result is considered the final grade of the learning component.

It is relevant to mention that this general structure of grading is not unique to ACS, but relatively common throughout other Romanian faculties. Inside UPB,

²<https://acs.pub.ro/>

we find the same system implement in every faculty [11].

Outside of UPB, other universities that have computer-science-related programs, such as the University of Bucharest [12], Bucharest University of Economic Studies [13], and The Technical University of Cluj-Napoca [14], follow the same pattern.

2.3 Tasks, Assignments, and Deadlines

As described previously, we will continue to refer to the graded activities held during the semesters as university tasks. For the purpose of this paper, we will focus on a specific category of tasks, namely assignments. We will analyze how they are defined inside ACS, but keeping in mind that the other faculties also use this structure.

Our main goal is to help students organize their schedule, and as homework, projects, and others span over multiple days and even weeks, we want to represent them inside our solution properly. Therefore, we need to find their particularities and how the students traditionally handle solving them. An important part of assignments is the date they must be solved and sent to be scored, which we will refer to as the deadline. A deadline can be of two types:

- A hard deadline, commonly known as the last day the assignment can be submitted to be graded.
- A soft deadline, the date until the assignment can be submitted to be graded without any score punishment.

In the period between the soft and hard deadline, daily penalties are usually applied to balance with the advantage of having extra time than required to solve. After the hard deadline, the assignment is considered closed, and there no more submissions are accepted.

Knowing this, we will be categorizing the tasks as follows :

- Homework: individual assignments, with a medium level of complexity, consisting of an apparent problem with its solution being correlated with a group of recently taught chapters in the corresponding course and further

exercised at the laboratories or seminars. It usually has between two to four weeks to be solved and submitted. This is the most common type of assignment, being used in most classes with a programming component or concept presented. Regularly, the final submission is in the form of a software solution that can be automatically tested locally or remotely to see if it correctly solves the problem.

- Project: individual or group-based assignments, with a more complex, more abstract theme that usually spans throughout the duration of the semester, with regular check-ins and a more subjective way of grading. They appear preponderantly in the latter years, as they test more skills and thus have more weight than other activities.
- Research: assignments with an extended period dedicated to researching a specific subject or concept.
- Tests: individual verifications.
- Administrative: a particular category of tasks for actions unrelated to the learning process, such as selecting optional classes, sending official documents, and registering for events.

2.4 Resources

Having described the core concepts that together define how the university is structured and how the educational process works, we will continue listing the leading platforms and applications that host the resources used by the academic community of ACS :

- University presentation websites³, mainly used for distributing general pieces of information to people from the exterior community.
- The faculty presentation website⁴ distributes available data and shares vital information for students, as the year structure, general-purpose announcements, timetables, and events.

³<https://upb.ro/en/>

⁴<https://acs.pub.ro/prezentare/despre-noi/>

- University platform dedicated to centralizing personal information for each student⁵, and also used a bridge for other administrative inquiries.
- Open courseware⁶, a public website used for publishing programming laboratories, homework, and projects.
- Moodle⁷, a web application based on enrollment to an attending class to gain access to its private materials, and also used for the examination process during the period of online learning in the Covid-19 pandemic.
- Assignment automatic checkers⁸, web applications used by the students to remotely submit their homework and run it against predefined data sets.
- Microsoft 365 for Education⁹ platform utilized as a support tool for remote activities in the period of online learning.

Other universities also have used some of the mentioned tools. The University of Bucharest has also implemented its own online student management tool¹⁰ and uses Moodle and Microsoft 365¹¹ for the same reasons stated above. It is also true for Bucharest University of Economic Studies¹².

Therefore, even though our solution is conceptualized for ACS, it can be adapted for other universities without changing the architecture and the relations between components.

⁵<https://studenti.pub.ro/>

⁶<https://ocw.cs.pub.ro/courses/>

⁷<https://curs.upb.ro/>

⁸<https://vmchecker.cs.pub.ro/ui/>

⁹<https://www.microsoft.com/ro-ro/education>

¹⁰<https://online.unibuc.ro/ghiduri/>

¹¹<https://online.unibuc.ro/alege-o-resursa/>

¹²<https://www.net.ase.ro/student/#>

Chapter 3

State of the art

As stated in the introduction, technology is in a constant process of discovery, innovation, and refinement. This brings a unique opportunity for developers to produce tools that simplify complex processes into digitally assisted components. One of these complex tasks, scheduling, is an ever-present point of interest for tech giants such as Google¹ and Apple², small companies, and finally, smaller groups of independent developers.

The main differences between all the applications on the market are for whom they are created, also known as the target audience, their price, and their popularity. Taking into consideration the developers and the comparisons between their products, in the following sections, we will be analyzing a few of the most used applications that accomplish the goal introduced in this paper. Then, focusing on the key functionalities and user experience, we want to determine what innovation they bring and what aspects can be improved.

As part of state-of-the-art, we will be dividing these productivity applications into two categories: general and specific.

We define an application that fits in the **general** category as a software solution with a broad target audience, having general functionalities easy to use to encourage user engagement and giving them a high level of customizability so that proficient users can genuinely use it at its maximum potential. These applications appeal to the consumers as large companies usually develop them with this as the primary target. Excellent examples for this category are Google

¹<https://about.google/>

²<https://www.apple.com/>

Calendar³, Apple Calendar⁴, and Microsoft Planner⁵.

The **specific** category comprises of applications developed with a specific user group in mind, which considers punctual problems and offers a more direct, straightforward solution. These applications focus more on refinement and dedicate more time and resources to finding the best solutions than expanding their reach. A good example of this are university-specific applications.

The critical difference between these two categories is the end goal of their applications. **General** applications are developed to appeal to people rather than simplifying the solution of an actual problem, letting the user free to use the tools available to create a personal solution.

Specific ones do it the other way around. They are created with the problem in mind, giving one optimal solution to all its users, expecting it to be adapted naturally by the people it was made for.

With this in mind, we will analyze state-of-the-art representative applications for both categories.

3.1 General productivity applications

The first application that we will be analyzing is Google Calendar, a general free solution for scheduling activities, developed to suit a broad audience, with over 11 years of existence [15] cumulating in over 500 million users. As part of the Google suite of applications, it is very accessible, installed by default in the Google ecosystem, such as Android devices⁶ and Chrome browser⁷, Google Calendar is a good point of start to determine what makes an application increase productivity.

To put this analysis into perspective, we will also draw parallels between them and their direct competitor, the default calendar application published by Apple. By doing so, we can better understand how big companies treat personalization and automation processes and deliver them to any type of user.

³<https://calendar.google.com/>

⁴<https://www.icloud.com/calendar>

⁵<https://tasks.office.com/>

⁶<https://www.android.com/>

⁷<https://www.google.com/chrome/>

3.1.1 Target Audience

When looking at the number of users who use products published by both Google and Apple, we can truly understand how big these two giants are. At the moment this paper was written, Google has over a billion people that use its products. Adding this to the facts that Google owns over 91% of the search engine market share [16] and 91% of all internet searches are made from a smartphone [17], where also Google dominates as Android is present in over 72% of all smartphones, then we can barely grasp how big Google has become. In comparison, Apple has an entirely different strategy, introducing an enclosed ecosystem, but with the same goal to get as many users to use their product. Apple integrates its software solutions only into proprietary hardware, meaning that its software is specifically made for its devices. At the same time, Google's Android is open source, meaning that any product that suffices the hardware requirements can use Android as their operating system under Google's rules.

These two strategies have a common trade-off: quality versus quantity. So in both cases, the target audience is the electronic device market, leaving it to the user to choose based on what they value or need the most. In either case, users usually preferred to use the default applications developed by these companies, included by default in their software, as they have years of refinement behind them.

With over billions of devices that use software produced by these companies, we can safely assess that both of their strategies worked. As both of them include planning functionalities, we are interested in their implementation and how their large target audience adopts it.

3.1.2 Functionalities

Regarding their primary functionality, both handle the most basic interactions regarding planning, which is creating a digital representation of a real-world event. They also send notifications, and most of the time, they are simply used to search a specific day of the month or year and put it into perspective. In table 3.1, we see a list of planning university activities functionalities that a student might find useful.

Table 3.1: **Calendar** relevant functionalities

Functionality	Google Calendar	Apple Calendar
Different types of events	✓	✓
Intuitive interface	✓	✓
Upcoming events list	✓	✓
Sharing calendar	✓	✓
Integration with other applications	✓	✓
Machine learning	✓	✓
Import/Export	✓	✓
Template for a task event	✗	✗
Statistics based on event data	✗	✗

We have the option to easily translate simple, all-day, recurring, and location-bound events into the calendar. These events can also be freely grouped. It also supports integration with other related software, such as email and teleworking applications, and additional custom input. Regarding integration, one important feature that Google's Calendar supports are imports from different sources, Apple's Calendar included. As users more and more adopt both providers and have to switch constantly between platforms, these types of features that help with the migration save considerable amounts of time.

Sharing a calendar is also an option, but only with known people. This is an inconvenience when we try to translate a calendar into a university scheduler, as the owner has to invite the other colleagues personally, and so other people might be left out or have to make a calendar themselves. Also, the lack of a template leads to a lack of a standard for creating an assignment, so different calendars might have different representations of the same event.

Also important to mention, as *artificial intelligence* starts to mature [18], its effects are beginning to show, as more calendars adopt features that predict users' input and guide them to more easily set up and give them suggestions based on their behavior.

3.1.3 UI & UX

From a user experience perspective, the interface is relatively simple but powerful enough so that the main points of interest are accessible right on the front page. The calendar itself is in the center of attention, and as the main point of focus, events related to it are directly shown above dates, giving a visual overview of the schedule. We remark how different events are presented based on their type, group, and user personalization.

This type of interface is an example of utilizing visual clues and simplifying rather long and convoluted pieces of information in a more streamlined way. Images extracted from these application can be found in appendix A.

3.1.4 Conclusion

Putting all this information into the perspective of an ACS student, whom we consider a proficient user, these applications have more than enough features available to help improve its productivity and automate many time-costly processes, but as mentioned in the Scope chapter, there is a significant overhead in translating all the specific university aspects into general use applications. Furthermore, this type of self-organizing behavior requires constantly modifying the calendar as university activities are dynamically established and changed, creating a new responsibility always to update the calendar.

We can conclude that constantly updating data on an application to increase automatization defeats the purpose of automatization. As good as the features regarding personalization and presentation are, we cannot still map an accurate one-to-one relation between university activities and general digital events. Therefore, there can be implemented better solutions for university productivity.

3.2 Specific univeristy applications

In this section, for the reasons presented above, we will look at university productivity applications, how widely they are adopted, and how much they simplify student's work.

As the support application on which we developed the enhancing module is part of this category, being implemented with ACS students in mind, we will

focus on this application's particularities, putting them into perspective against similar applications. To better understand the starting point, we will continue to present the structure and timetable-related functionalities of ACS PUB Mobile before the additions of the features described in the introduction of this paper.

3.2.1 Functionality

The app is written in Flutter, Google's SDK for cross-platform applications. It is kept at the highest standards by manual and automated tests and checks, having the same quality of implementation as any big company product. First, we need to take a look at the state of the application, taken from the project's repository⁸.



Figure 3.1: Home page

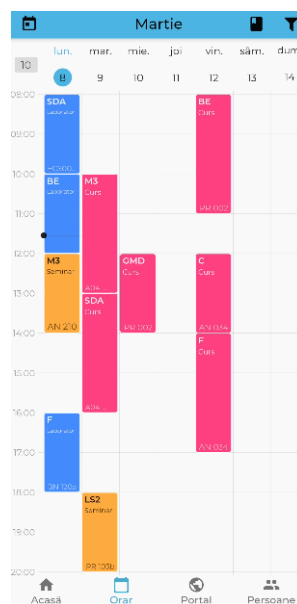


Figure 3.2: Timetable page



Figure 3.3: Filter page

As seen in figure 3.1, the application's home page contains elements related to news, but lacks planning-related elements, leaving space for development that we will explore in the following chapters.

As we shift focus towards figure 3.2, we find our main point of focus, namely the calendar page. We can see implemented simple and recurring events; both types also present inside Google's Calendar.

⁸<https://github.com/student-hub/acs-upb-mobile>

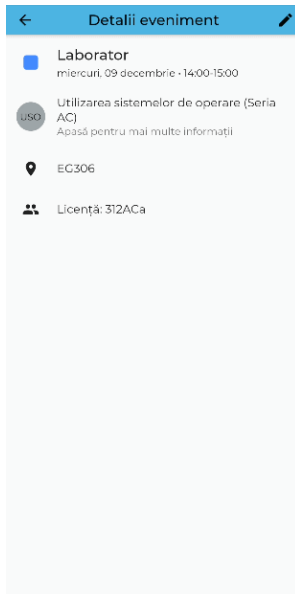


Figure 3.4: Event details page

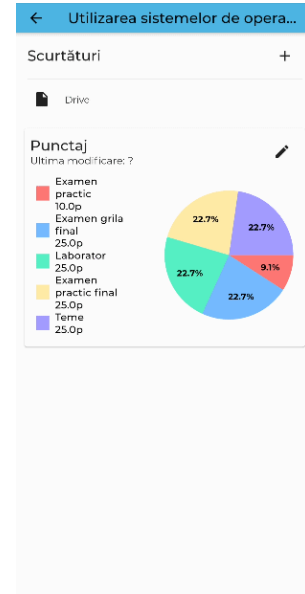


Figure 3.5: Class details page

The most significant advantage that ACS UPB Mobile brings is that by using a filter, like in figure 3.3, a user has instant access to his timetable without setting it up himself. This works because only one student has to introduce the events into the timetable, and these events propagate to those who choose to follow them. It is a simple and elegant solution to always having events updated for everyone, having just one person manage it.

As we can further observe in figure 3.4, when inspecting an event, it displays a list of information, such as the class linked to that event, time and date, location, and for whom it is addressed mainly, to aid the student. The course itself, shown in figure 3.5, also has a representation as it is used to give a more comprehensive perspective.

For our implementation, we need to just mention other existing features that we will be interacting with, as we are concerned with their functionalities rather than implementation:

- Users, as they are the main actor that interacts with all of the resources. We especially want to correlate users with timetable preferences without drastically changing their BLoC and data layer.
- Filters, that mimic the structure of the ACS curricula and lets the users

customize their interests.

- Localization, as the application comes in different languages.

3.2.2 User engagement

Analyzing figure 3.6, taken from the Firebase⁹ of ACS UPB Mobile, we can observe two big spikes around October in 2020 and March in 2021 that coincide with the beginning of each semester, giving us an insight into how the students use the app. We can presume that having a visual representation in the first weeks of each semester of the university schedule can significantly help memorize the events and not rely upon any calendar to optimize time management further.

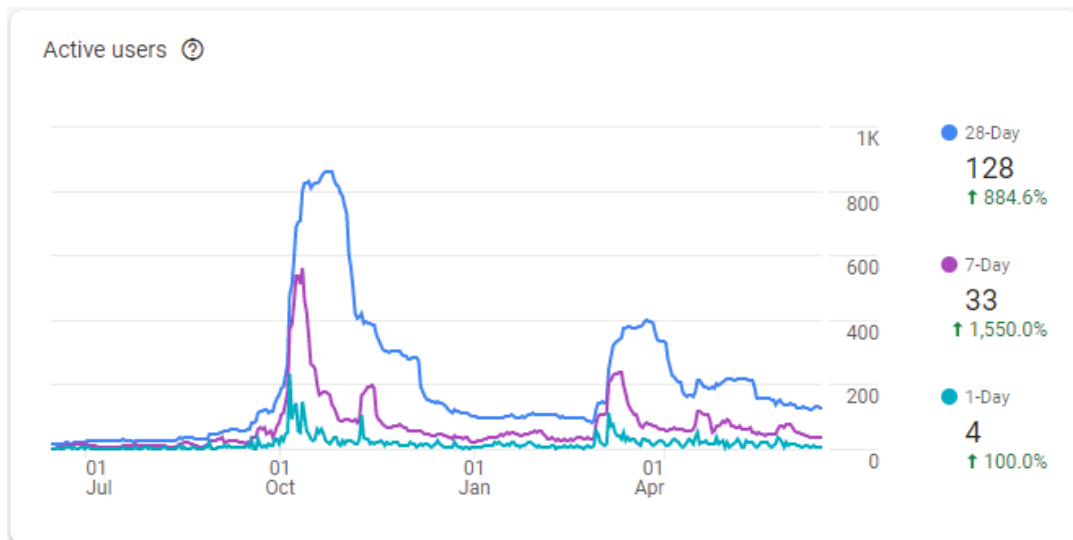


Figure 3.6: Active users

While proving that the application is known among more than 800 different ACS students, these spikes are a starting point for the application's success.

3.2.3 Conclusion

As presented in the anterior subsections, a specific application for handling scheduling university activities can gain spikes of users in short amounts of time.

⁹<https://firebase.google.com/>

Furthermore, as the community based around the faculty is tightly bounded, ACS UPB Mobile can further increase its popularity.

By adding a wide variety of productivity features and automating repetitive tasks, the spikes presented in the figure 3.6 provide a potential for retention, as users can freely experiment with the application and adopt it as a daily utility tool.

3.3 Technology

To properly understand the technicalities behind the application, we first need to describe the SDK used in the development process.

Flutter¹⁰ is a Google open-source software development kit used to create user interfaces for cross-platform applications significantly faster. It is written in Dart¹¹, which is a programming language belonging also to Google.

To simply describe Dart, it resembles Java's compilation process [19], as it uses a Dart virtual machine, capable of just in time compilation, meaning the code can be updated as it runs, without having to stop and restart the application. In addition, it is object-oriented and supports both stateful and stateless components.

Coming back to Flutter, the advantages of the programming languages translate into faster development, as compilation doesn't stop the running process, and it also saves the state in unmodified components. Flutter's philosophy is writing code only one time and having it compiled and ran on any platform [20]. This is possible because it doesn't use any native components but instead uses the Skia graphic engine¹². With this approach, the user can use an interface to create widgets, and the graphic engine handles the translation.

Widgets are the conceptual building block of any Flutter application. They are versatile components used to create a visual interface. Together with the triggerable events, they compose the user interface and only deal with the application's presentation layer. Therefore, widgets should only be a bridge between the users and the services.

¹⁰<https://flutter.dev/>

¹¹<https://dart.dev/>

¹²<https://skia.org/>

This layered pattern is called BLoC, abbreviated from Business Logic Component [21]. It is a Three-tier architecture [22] that links the user interface to a controller layer that handles the logic and sends data to the model. The controller layer, also known as the BLoC layer, acts as a bidirectional channel and controls the separation of logic between conceptually different components. It is based on reactive programming and it is designed to work only with streams of data [23].

The last concept we will present is the state. As widgets can be of two types: Stateless and Stateful, we need to understand when to use one or the other:

- A Stateful widget is usually a dynamic component, suffering changes while the users interact with it. The state itself is a general mechanism for checking that the application behaves in predictable ways, even with unpredictable input.

- A Stateless widget is used in the opposite case, as the component doesn't suffer any changes between renders. This doesn't mean that the data never changes, but rather it is influenced by other processes triggers and it updates specifically when they occur.

Like React¹³, Flutter has one advantage: its components are structured in a *tree* form, with pages as the roots for integrating widgets. In addition, the widgets can incorporate other widgets, referred to as children. This arborescent structure is vital to the rendering mechanism, as when a branch changes, it only affects the elements on that branch, so much performance is gained by not heaving to re-render the whole tree [24]. In this structure, it is recommended that Stateless Widgets are leafs, meaning nodes without any children, and have a Stateful parent, so when a change happens, it is propagated to the children and forces a render.

¹³<https://reactjs.org/>

Chapter 4

Interface

In this chapter, we will be discussing the presentation layer of the proposed solution of this paper. We will present the design conventions and strategies, the timetable widget, and the new elements and flows.

We will focus on the User Interface (UI) elements and how they are interconnected and behave, defining the User Experience (UX).

4.1 Design conventions and strategies

As the application is cross-platform, it was primarily designed to behave the same on all devices. However, as Android and iOS applications are conceptually different, ACS UPB Mobile borrows design conventions from both environments [25] and delivers a coherent interface and experience. As we previously mentioned, widgets play an essential part in the interface. As the Skia engine renders them, we are guaranteed that the user flows remain the same, while the composing graphical elements might differ. These differences are related to the operating system primitives called upon when building a page or a widget, and be can, for example, the top bar, pop-ups, and alerts.

The primary elements that we will use will be Cards¹ and Lists². They are already heavily used inside the application, versatile enough to contain various children elements, and interactive to respond to user input.

¹<https://api.flutter.dev/flutter/material/Card-class.html>

²<https://flutter.dev/docs/cookbook/lists/mixed-list>

The pages that we will update or add are :

Homepage:

- A new Card that displays pieces of information about upcoming events.
- A new Card for the Planner feature that displays information about the following tasks: it also doubles as the entry point for the Planner page.

Timetable:

- A Gantt diagram [26] way of displaying tasks.

Add/Edit task event:

- Add new options for adding tasks, with relevant fields.

Class:

- A new card to show all the events.

Event Instance:

- Button to hide/unhide event.
- List tiles for task-specific fields.
- Button to add/edit the goal.
- Cards for each element of the goal.

Planner Page:

- An effort graph.
- Lists for current, hidden, and past events.

4.2 Timetable

The timetable is based upon the public package: `timetable: 0.2.9`³, which is another open-source project created by a student⁴ from Hasso-Plattner-Institute⁵, a university in Potsdam, Germany.

As we can see in figure 4.1, taken directly from the application, we can split the view into three main zones of interest :

- Date header
- All-day events bar
- Class events

For our implementation, in which the tasks are all-day events, we will use the all-day events bar to display them. The main benefit is that it generates a simplified Gantt chart of the tasks the users have to complete, and it also visually distinguishes them from other types of events⁶.

Adding a task follows the same process of adding any other event, as it requires the user to preferably tap on the day in which the event starts, choose the corresponding type, and complete the other fields. Editing them works exactly like editing any other event.



Figure 4.1 Screenshot of the timetable

³<https://pub.dev/packages/timetable>

⁴<https://github.com/JonasWanke>

⁵<https://hpi.de/>

⁶<https://github.com/JonasWanke/timetable>

4.3 Red Routes

We define the Red Routes as the critical functionalities that bring the most value for our users. To highlight them, we have to look at the usability matrix for all the proposed functionalities presented in figure 4.2.

Usability Matrix				
Used always		Checking goals	Checking Timetable	Checking events on homepage
Used Frequently			Updating a goal	Checking a task page
Used occasionally	Editing a Task	Checking all events of a class	Hide an task	Checking Planner page
Used Rarely	Adding a Task			
	Used by a few	Used by some	Used by many	Used by all

Figure 4.2: Red Routes matrix

When making the matrix, we analyzed each functionality by two factors: the number of users and the frequency of being used. Presented in red background, those functionalities are critical for the purpose of optimizing the university schedule, as they are used by a majority of the users frequently. On the other hand, even though not very used, the functionalities in the lower part of the main diagonal are vital for the other features, as they deal with creating the events used by more user-centered features.

As the figure indicates, the most important functionalities for the users are those that quickly give them information about events and tasks in the near future. This provides them with a better understanding of how time is partitioned and how it should be managed to accomplish everything that is tracked.

4.4 Wireframes

As every visual update has to be coherent with the other components, we want to first conceptualize new elements by including them in new wireframes. By doing so, we can have a general view of how they should be positioned. This way, we ensure that the user recognizes the new features and quickly understands and uses them.

Based on the features that we want to add, we identified the pages that should contain them, and for those that need a new page, we choose the best way to integrate a new flow in the application:

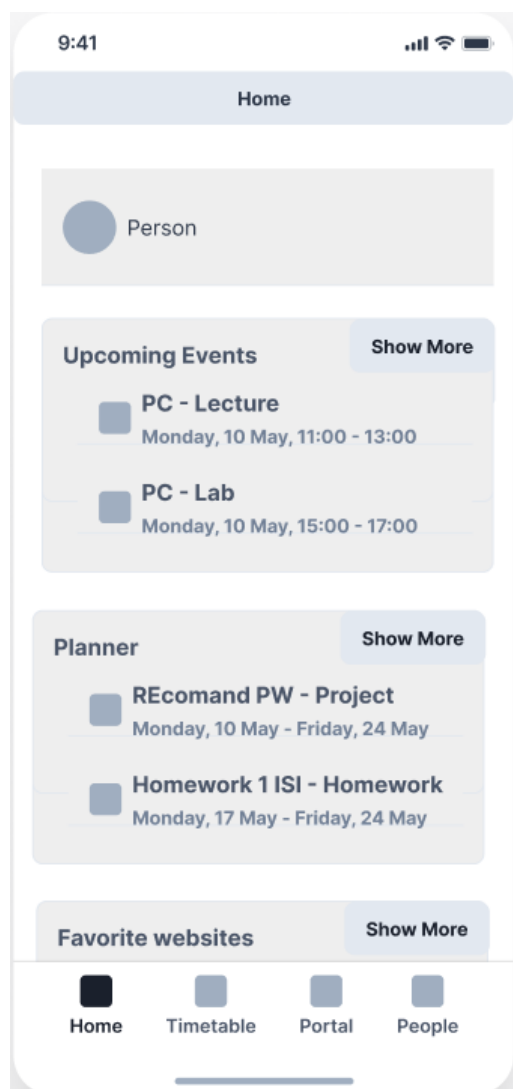


Figure 4.3: Home page

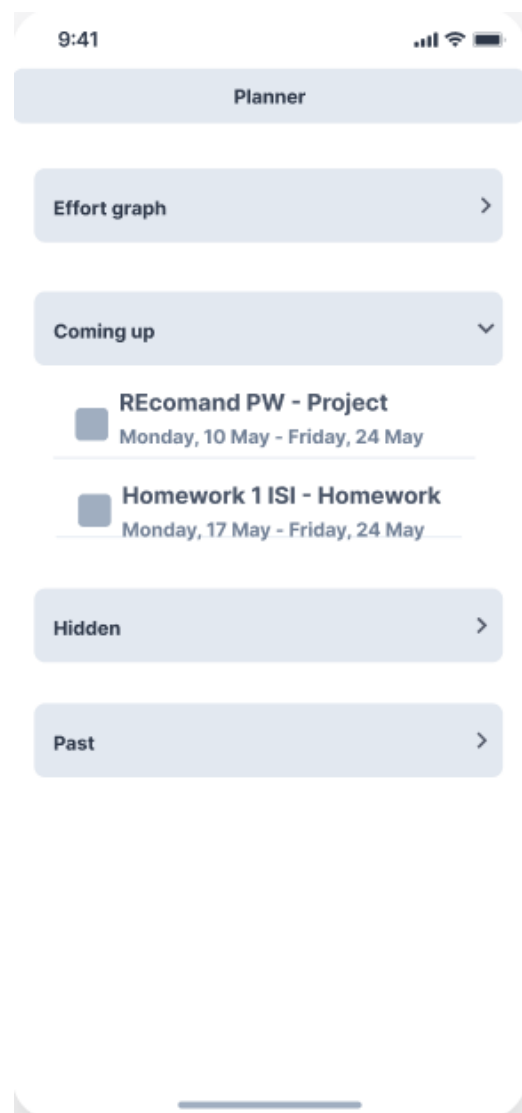


Figure 4.4: Planner page

Homepage (figure 4.3) has conceptually integrated within the new Cards. In Upcoming events we have a short list of upcoming class events. For each event, we show the class abbreviation, type, and time frame. The *Show More* button redirects to the timetable. The Planner card follows the same logic, showing for each element the name, class abbreviation, type, and the start and end date. The difference is *Show More* redirects to the the Planner page (fig. 4.4).

The Planner has is divided in four sections. The first is reserved for the distribution graph of tasks throughout the semester. The others contain tasks that comply with the category suggested in the title. Tapping on the title shows or hides the list of containing items.

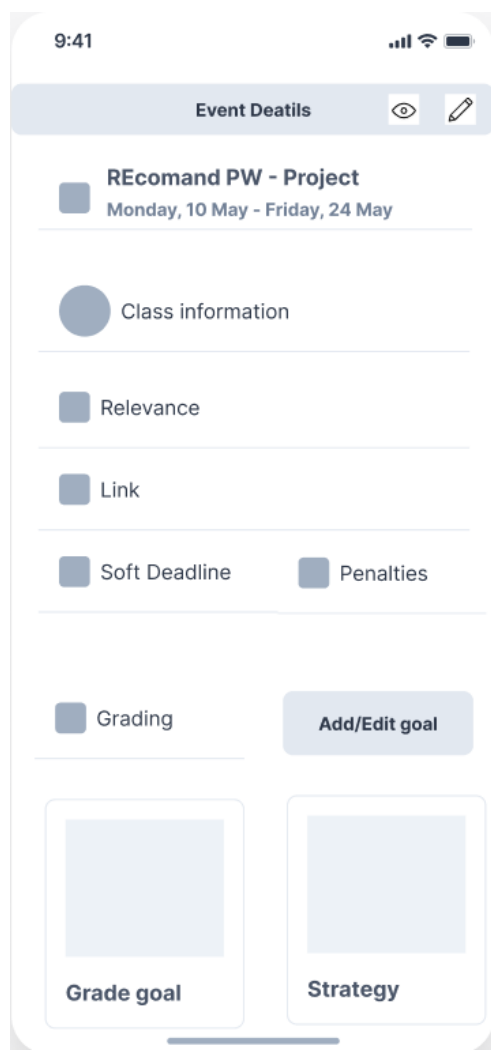


Figure 4.5: Event Details page

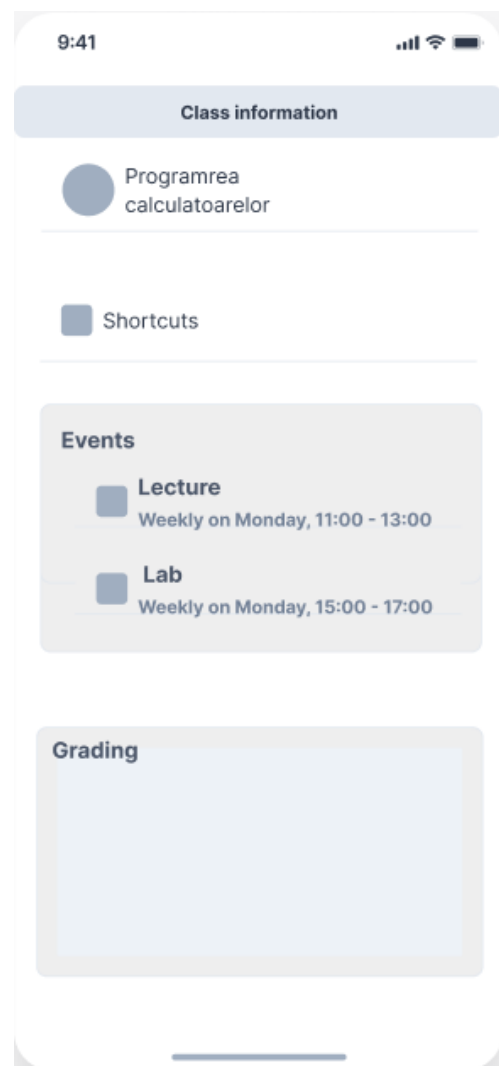


Figure 4.6: Class Information page

Tapping on an task, it redirects to the event details page (figure 4.5). Here we have all the fields specific to tasks and also the goal button, and the visibility and edit button. For an event that already has set goals, like the one in the figure (figure 4.5), it also displays them in separate cards.

In the class information page (figure 4.6), we have added a new Card to display all it's all events in just one place.

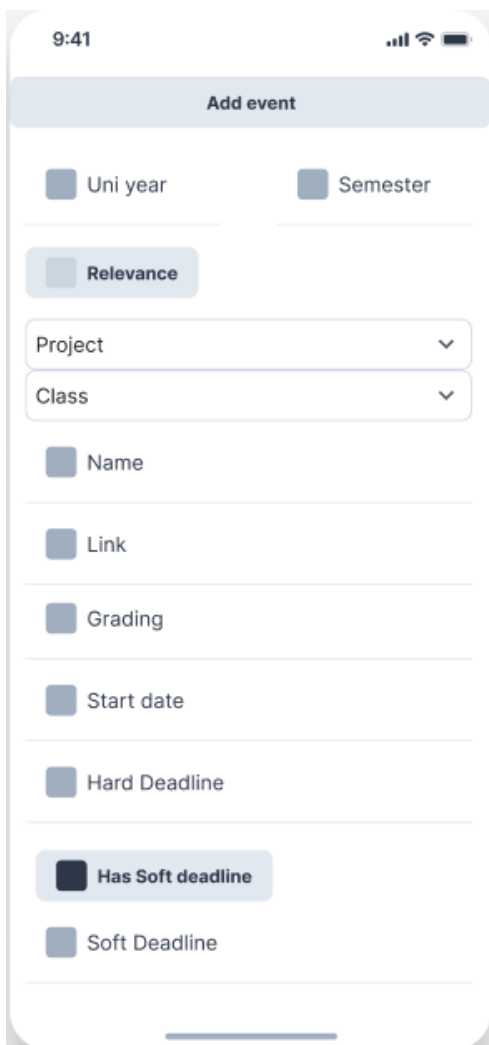


Figure 4.7: Add/Edit event page

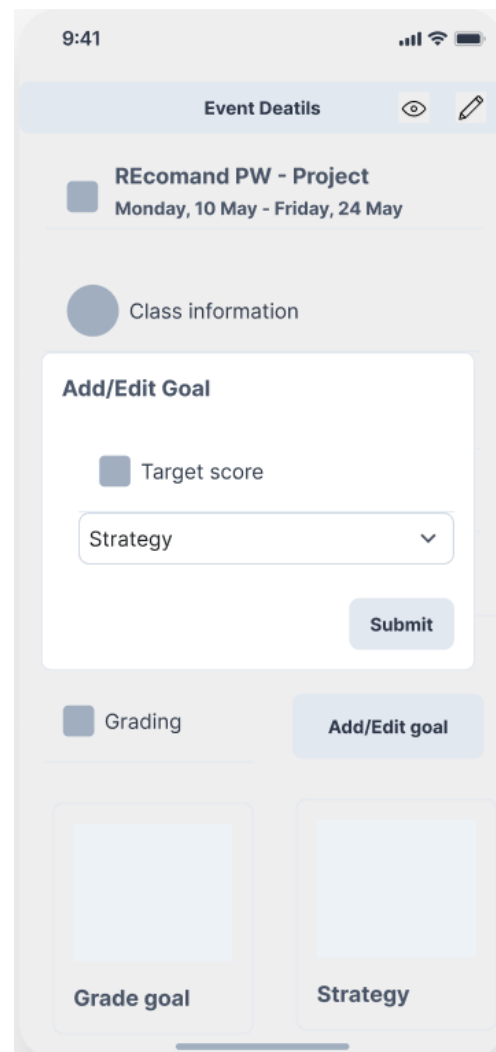


Figure 4.8: Add/Edit goal

In figures 4.7 and 4.8, we have the new form for adding or editing an task in a separate page, respectively an goal in a pop-up.

4.5 Final UI

After we established the wireframes, we now will present the final implementation of Homepage (fig. 4.9), Planner (fig. 4.10), Timetable (fig. 4.11), Event Details page (fig. 4.12), Class Information page (fig. 4.13), Add/Edit event page (fig. 4.14), and Add/Edit goal pop-up (fig. 4.15).

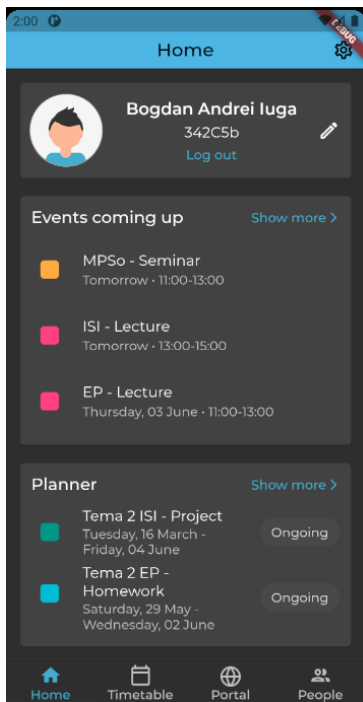


Figure 4.9: Homepage

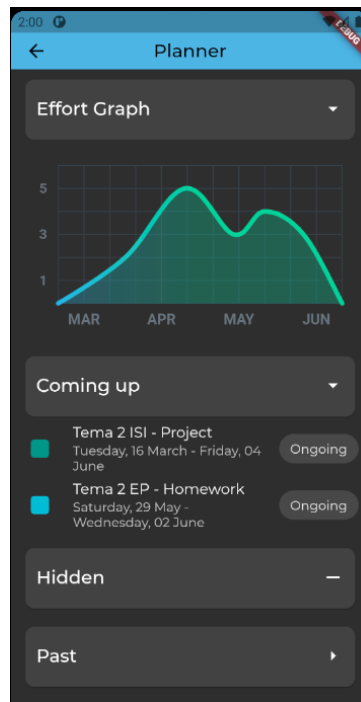


Figure 4.10: Planner



Figure 4.11: Timetable

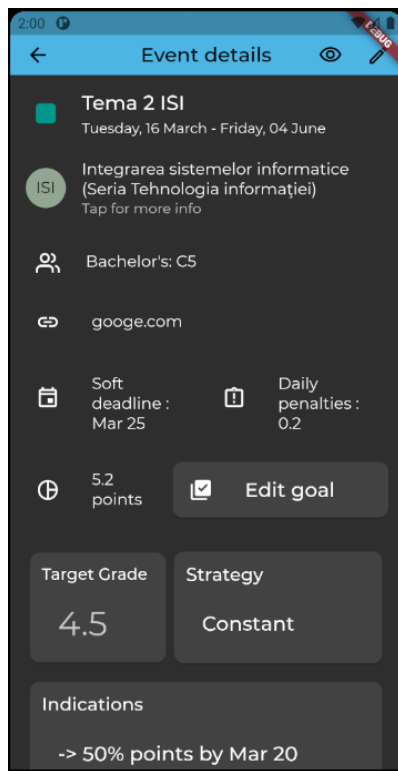


Figure 4.12: Event page

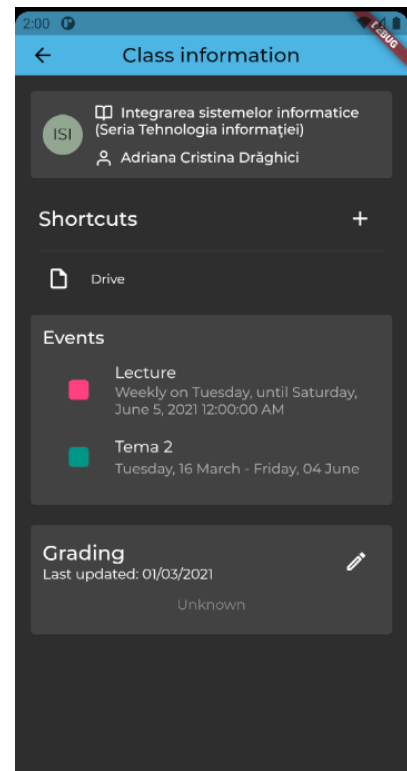


Figure 4.13: Class page

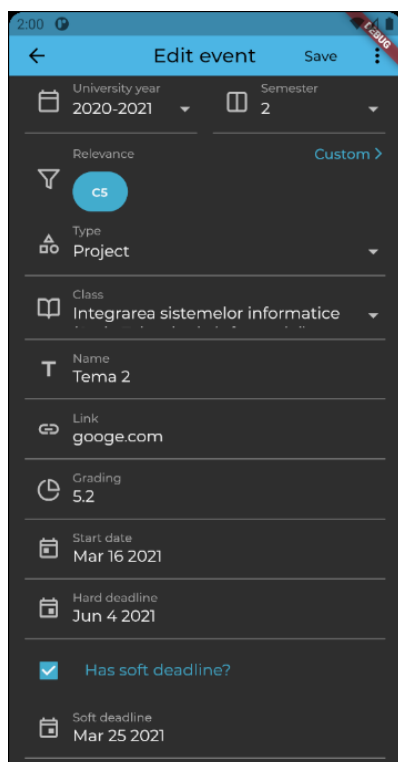


Figure 4.14: Add/Edit event

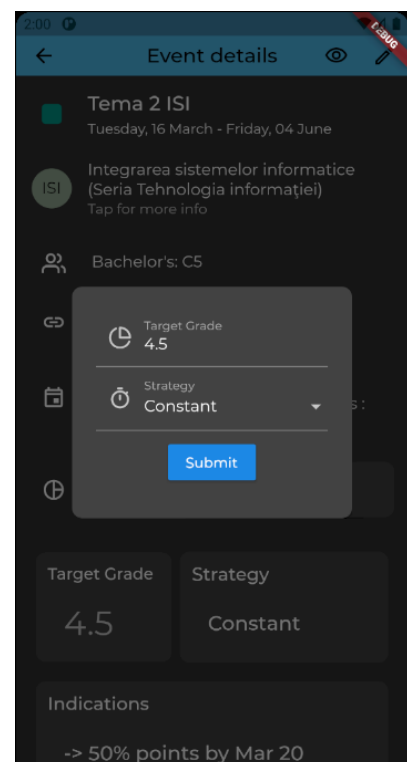


Figure 4.15: Add/Edit goal

Chapter 5

Implementation

This chapter will analyze ACS UPB Mobile from an architectural perspective, describing the framework based on its inner workings, data models, and design patterns. We will explain how we conceptualized and integrated the solution proposed in this paper to fully integrate with the codebase already existent and add further possibilities for improvement.

5.1 ACS UPB Mobile architecture

5.1.1 Database

As every software solution needs a non-volatile storage environment, the application uses Firestore¹, another component of the Google ecosystem. It is a modern cloud-based non-relational database designed to work with other Google SDKs, such as Flutter. As a document-oriented database, its structure is based on two main components: documents and collections. Documents are files that contain the application data in a JSON-like format.

Each document has a unique identifier and is composed of a list of pairs in the form of **keys** and **data**. A key is a unique string, and data can be of primitive and complex data types², such as null, boolean, integer and double, date, string, and also Cloud Firestore reference, Array, and Map. A special feature is that documents don't have a predefined format, so their structure can be easily manipulated. Collections are groups of documents that have a logical

¹<https://firebase.google.com/docs/firestore/>

²<https://firebase.google.com/docs/firestore/manage-data/data-types>

correlation with each other. As mentioned above, they don't have to respect a strict structure, giving a significant level of liberty in implementation.

One example that proves the advantage of using Flutter, which is also of interest for the purpose of this paper, is the database structure of university events. As we will further elaborate on these **events**, for the moment, it suffices to know that, in the database, we use the **event** collection to save all the documents related to events. As we explained in an anterior chapter, in the application, we have different types of university events that require in the related document various fields. For instance, while an **All day event** requires just an end date, a **Recurring event** needs a recurrence rule³. Instances of these events are saved into documents in the same collection and can be retrieved separately by doing queries that check the existence of a key-data pair in them.

While an old approach would be either creating different tables for these events or creating only one containing all the fields, it would raise many scalabilities and technical difficulties as data integrity checks and queries become complex. This solution is easily scalable, secure, and extensible⁴, and in the next section, we will present how we integrated new events and collections.

5.1.2 Provider

As we remarked above, State management is vital to any Flutter application and can be done in different ways. The default way is using the `InheritedWidget`⁵, a class that propagates changes down the components tree. In ACS UPB Mobile, the State is managed with *Provider*⁶ and *ChangeNotifier*⁷ classes, which are wrappers over `InheritedWidget` that significantly reduce the boiler-plate code and application complexity.

A Provider's core philosophy is being defined at the top of the branch that requires it, as it can be called from inner nodes. In the application, we define all the providers at the root of the application so that we can use them where they are needed. Appropriately named providers, they are composed of asynchronous

³<https://datatracker.ietf.org/doc/html/rfc5545#section-3.3.10>

⁴<https://firebase.google.com/docs/firestore/security/rules-structure>

⁵<https://api.flutter.dev/flutter/widgets/InheritedWidget-class.html>

⁶<https://pub.dev/packages/provider>

⁷<https://api.flutter.dev/flutter/foundation/ChangeNotifier-class.html>

methods that mainly interact with data transfers between the application and database.

These asynchronous methods usually return a **Future** that behaves similarly to promises from Javascript⁸. These Futures guarantee that the called method will finish, be it in an unknown time, and some data will be returned while the application continues to run. This type of behavior is vital, as it keeps the application running for the users while internally it waits for the transfer of data.

With providers defined at the top level, we want a way to update a branch of the application without refreshing the whole tree. This is where `ChangeNotifier` is used, as it restricts the refreshing behavior. Together with `Provider`, it assures that when a change is made, it notifies and propagates to all the branches that use it, so only they have to modify.

Using this behavior, we can easily control the application tree's State throughout the application tree while doing it in an optimal way.

5.1.3 BLoC Design Pattern

Knowing now that the providers are being used to transfer data between the user and the storage environment, we can present all the layers of the application. The application respects the BLoC design pattern presented earlier, as all the components use this three tyre approach.

As we can see for example in figure 5.1, the timetable page is divided into three components :

- Model, that is the equivalent of the data layer, where we can find the class definitions for the objects used inside the application.
- Service, which can be considered the BLoC layer that handles the data transfers and communication between layers.

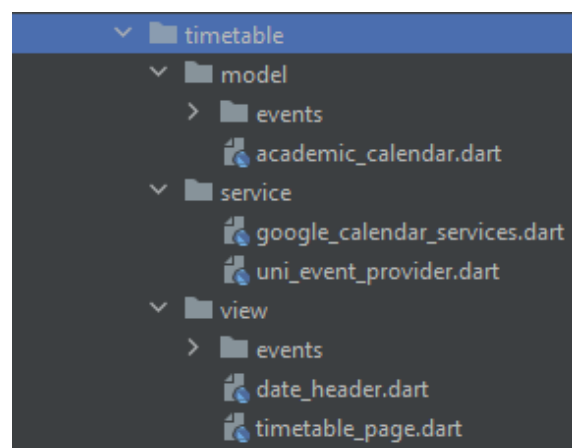


Figure 5.1

⁸<https://www.javascript.com/>

- View, known as the presentation layer in this architecture, is where the code responsible for the interface can be found.

5.1.4 Flow

We will now explain a general flow of the timetable component of the application concerning the design architecture.

As we first access the timetable page, the View components start the initialization process. By having access to the provider, they ask the BLoC layer for data and receive a Future object, so the application doesn't block while data is being retrieved. As the raw data is gathered from the database, it is then propagated to the data layer, where the object's constructors can be found. When the object instances have been created, they are sent back to the BLoC layer to be delivered to the presentation layer. Finally, when the Future is solved, the view is complete, and the final data is presented to the user.

This flow will also apply to features that we will implement for this paper's purpose.

5.2 Integration strategy

In this section, we will describe the integration strategy of the proposed solution, as we want to follow the development principles and conventions that apply to ACS UPB Mobile. Going back to the proposed functionalities enumerated in the Introduction chapter, we will punctually explain how each had to be integrated.

The first proposed functionality was the integration of assignments as choosable event types for users. This implied the need for a new class for them, as we need additional data to translate a real-world university assignment into a digital representation correctly. Furthermore, these events have to be in the same collection as the other events and be retrieved and handled accordingly. We also need to address the creation, read, update, deletion as the new fields imply new data transfers checks.

The second proposed functionality deals with the overall planning of the events that the user wants to track. As a filter retrieves all the class events, a user

might not want to see all of them, so we have to give them a more granular option that applies only to him. Creating a “hide” option for an event allows the user to choose what events he wants to see in his planner. Other widgets will be built for a more simplified way of actually seeing assignments and tasks.

The third proposed functionality deals with assisting the user in the solving process, as we want to give him more visibility over his work and advice over the implementation strategy. We want to create a new collection for these “goals”, where we can store data for each user, as this implementation leaves a possibility for new features like teams of students tracking one project and setting a common goal that each can follow.

The fourth proposed functionality refers to statistics, and these can be generated based on the already available information. We need to decide what metrics are relevant and how to visually present them so the user can quickly understand the information. This process is only handled in the presentation layer, so no further integration is needed.

The final goal is to integrate all that we proposed in the paper. This can be verified by using the GitHub platform to monitor the changes. In a following chapter, we will further expand the delivery process.

5.3 Database integration

As we have access to the development database, we can directly interact with it, as we follow a basic guideline. Because our solution doesn’t change the flow of other components, we don’t want to modify already existing fields or structures, but to add new fields that help our new features. We will enumerate the collections that have been changed or added and present their purpose.

events

The events collection contains all of the academic events that the users have added. In the following table (5.1), we will present the structure of an task in the database. The elements changed or added in the development process of the solution described in this paper are highlighted with the color yellow.

Table 5.1: **Task** event structure

Field	Type	Information	Example
type	string	one of "homework", "project", "test", "administrative", "research"	"homework"
addedBy	string	Id of the user that added it	"ti0QcVZFjZCtj50WW6"
calendar	string	The event must be correlated with an academic year	"2020"
class	string	Name of the class	"L-A4-S2-PW-Tehnologia informației"
degree	string	Correlated learning cycle	"BSc" (bachelor's)
relevance	array<string>	Filter relevance	["C5", "C1"]
name	string	Name of the event	"Tema 1"
duration	map<string, number>	Explicit duration	{ "days": 15, "months": 2 }
location	string	Link where the assignment can be found	"https://ocw.cs.pub.ro/courses/pw/tema1"
editable	boolean	If the event can be edited	true
start	timestamp	Start date of the event. It must have a value	14 May 2021 at 00:00:00 UTC+3
softDeadline	timestamp	Soft deadline of the event	24 May 2021 at 00:00:00 UTC+3
hardDeadline	timestamp	Hard deadline of the event	28 May 2021 at 00:00:00 UTC+3
grade	number	The maximum grade of the event	4.5
penalties	number	Daily penalty points after the soft deadline	0.1

As we can see in table 5.1, we had to add new event types, update the location field to accept links, and add new fields relevant only to tasks. Deadlines represent precisely what they are named after, and we save penalties and grades inside the event. Finally, we use `editable` to add another layer of security over events, as maybe some of them should not be modified.

users

In the “users” collection, we find documents that contain user-specific information, such as classes and filters. However, as our new event types respect the general structure, they are already integrated with the filtering mechanism.

Table 5.2: **Users** new field

Field	Type	Information	Example
<code>permissionLevel</code>	<code>number</code>	Permissions that the user has	4
...			
<code>hiddenEvents</code>	<code>array<string></code>	List of id's of hidden events	[“RN5jsfBHgDb”, “Reavx6YOEHSJJ”]

For the purpose of the new features, we only need to look at the new field added: **`hiddenEvents`** (table 5.2), which is a list of the identifiers of the tasks that the users want to hide.

goals

A new collection that consists of documents that, for each user, remember what goals it has set for its events.

The structure of the document(table 5.3) :

Table 5.3: **Goals** structure

Field	Type	Information	Example
<code>addedBy</code>	<code>string</code>	Id of owner has	“ti0QcVZFj ZCtj50WW6”
<code>taskGoals</code>	<code>array<map></code>	Each element of the array represent a goal for a task	table 5.4

Table 5.4: **taskGoals** structure

Field	Type	Information	Example
taskId	string	Id of related task	"RN5jsfBHgDb"
targetGrade	number	The target grade	4.5
strategy	string	one of : "constant", "early", "late"	"constant"

By using this structure, we can independently store information about the user's targets. Its main advantage is that we don't expose more information necessary, leaving a place for collaborative tasks to be implemented.

The fields in table 5.4 are **targetGrade**, for establishing a goal to reach, and **strategy**, to establish how it can be reached. At the moment, we have three predefined strategies, giving advice on how reach the goal in with constant effort, or with more effort either earlier or later in regards to deadlines.

5.4 System integration

Concerning our solution, in this section, we will analyze the BLoC and Data layers implementation.

We will break it down into two parts, methods that use already existing data and methods that use the new collections.

In the first part, we are interested in integrating a new type of event in the already existing pool of events. For this, we had to understand the mechanism of how the raw fetched data from the "events" collection is handled.

The general flow for events is :

- The collection is retrieved from the database.
- We iterate through the documents and send them to the data layer so the corresponding events are created. In this step, we have to add a differentiator for tasks, and we do that by checking if the document retrieved has a **hardDeadline** field set, as it's required and unique to only **tasks**.
- A stream of events is available to be consumed by other methods in the provider.

As we mentioned previously, for every event type, there is a data model in the project. The one that was of interest to us is the **AllDayUniEvent**, as tasks are frequently spread throughout multiple days and end at midnight.

Knowing this, we can consider the **AllDayUniEvent** as a parent class for our **TaskEvent**. Furthermore, **AllDayUniEvent** is a direct child of **UniEvent**, the general implementation of events inside the application. Therefore tasks are recognized as university events. One adaptation was required, as the **hardDeadline** for the task had to become the **endDate** in the **AllDayUniEvent** constructor, because both parameters are required. Besides this, the other specific data pairs described in the previous section were saved as instance fields.

Now that we have task events and other events, the next step is to create appropriate instances of these events. Every event has to have a generator of instances, as some events are recurring, and each instance has to exist independently. As tasks are not periodic, one task event generates one task event instance, creating an easily trackable one-to-one relationship between the two.

Using this information, we created new methods in the events provider, most of which deal with selecting events from a date interval, events of a specific class, specific task.

For example, as the mechanism works on Futures, once the events were selected, their instances were generated and then returned to the pages that need them. The same principles apply to the user provider when we update the hidden events list.

In the second part, we will analyze the process of creating the planner feature, ignoring the presentation layer for the moment, as it will be detailed in the next chapter.

For the provider, we need to register it on the root of the application tree. We do that by using the **Multi Provider** functionality, a list of multiple providers created at the start of the application, from the provider package. Having it registered, we can now use the provider inside the application.

The planner provider consists of methods that mainly deal with customization of tasks solving processes. It is only natural that we don't alter the structure of events, so we have to create a new data model named *Goal*. As presented above in database integration steps, a goal will follow the table 5.3 structure.

We implemented the basic **CRUD**(create, read, update, delete) operations for goals inside the planner provider and conversions between data objects fetched from the database and instances to assure the integrity of the data transfer. Using these asynchronous methods, with others interacting with events visibility, we can solve complex processes with reduced effort.

We will describe the flow of operations from the BLoC and Data layers perspective, as the user wants to add a new goal for an event instance :

- The user opens the page of an instance of a task.
- The document that contains his goals is fetched from the database. A list of goals is created from the taskGoals field, and if that list includes a goal for the event that the user is interested in.
- If the document doesn't exist or the goal is not present in the list, a null value is returned, signaling that a goal is not set for the said event.
- If the goal is not set yet, it adds a new goal for the task. The application checks the collection to find the user's document, and if not, creates one with an array of taskGoals of one element, the one that it had to be added.
- If the document exists, the taskGoals list is fetched, the new event is added to it, and then saved in the database.
- If the event is already saved, now the user can simply update them.

With this flow, we can control the planner feature and help the user better schedule his assignments while effortlessly creating and updating them for all the users that follow the same classes. Diagrams for this process can be found in the appendix B.

Chapter 6

Contribution process

ACS UPB Mobile is hosted on GitHub¹, an internet solution for managing software versioning using Git². As an open-source application, the project's repository is open to everybody under the MIT license³, and this allows many developers to use or contribute to it.

For the purpose of this paper, we will explain how we adhered to the development process of the application using the tools provided by GitHub. We will present the steps involved, from proposing a feature to it being implemented, verified, and deployed.

6.1 Repository

The ACS UPB Mobile repository is the main point of interest for us. Here we can find all the code versions, development guides, changelogs, and contact information. To ensure that any proposed feature is relevant to the application, the first step is to discuss with the project owner and reach an agreement about its implementation. We can break this discussion for any features into three points that need to be clarified :

- How it will affect the architecture and other components
- How it will affect the user interface and experience

¹<https://github.com/>

²<https://git-scm.com/>

³<https://opensource.org/licenses/MIT>

- What opportunities for future features it enables

For the features proposed in this paper, we already answered these questions in the previous chapters. As this discussion was the first step in the implementation process, we had a clear development roadmap, as we knew ahead of time what challenges and opportunities each new feature provides.

In order to keep track of the progress made, we created an issue inside the repository for each new feature. To further separate these related issues from the others, we also created a project for the general module.

This structure is illustrated in figure 6.1 ⁴.

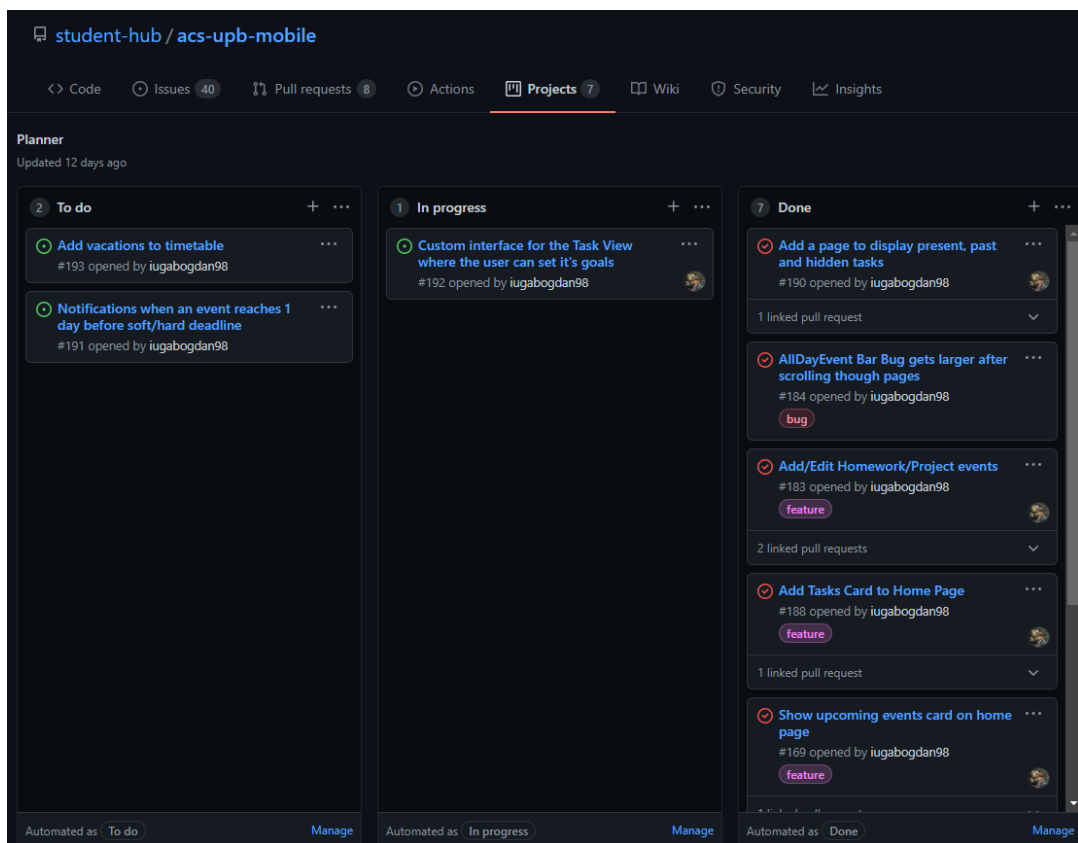


Figure 6.1: Planner project

Furthermore, because we were using Git, each of these issues had its own separate branch, meaning that the code was isolated while in development, assuring that we don't change anything until the issue is resolved.

⁴<https://github.com/student-hub/acs-upb-mobile/projects/6>

6.2 Pull Request

When a feature is fully implemented, the next step is to add it to the project's codebase, which is found on the "master" branch. As a security measure, merging code to the master branch has to meet a series of checks and approvals, and it is formally made by creating a pull request.

Before creating a pull request inside the application repository, we had to ensure that the following list of code quality checks is satisfied:

- The coding style corresponds to Dart standards.
- Dart analysis returns no warnings or errors.
- Each new text added is localized.
- All the automated tests are passed.

As some features may change the application, it is crucial that all the tests pass so that we know that even with a new feature, the application still behaves as intended. A concrete example of having to update the tests after developing a feature is that, when adding a new provider inside a page, it also has to be added in all the tests that use the page, otherwise they fail. In this way, we can check if the behavior of the new component produces a different result than the one expected. Github also automatically checks each pull request to ensure that the quality required is met.

The next step is requesting a code review, where in our case, the project owner manually checks the solution and decides if it is good to be merged. In the case of a negative response, further discussions happen where it is explained what is needed to be changed, and the process repeats until a positive response is given.

6.3 Deployment

ACS UPB Mobile is, at the moment when this paper was written, available in the Google Play Store.

When a Pull request is approved and merged to the master branch, the code changes only in the repository. As we want to deliver these changes to the users, we need to deploy the application. This time of deployment and version number

is established with the project owner and requires a list of changes that will be displayed to the users.

The deployment process is automated, and it is based on the tag mechanism used by Github. When pushing a new tag to the origin, the application is automatically deployed, and the code is checked automatically by the Play Store. When approved, it becomes available to the users.

In the development process of the module proposed in this paper, we followed these steps and managed to deploy features that are now used inside the application.

Chapter 7

Conclusion

7.1 Achievements

As mentioned in the first chapter, this paper proposed a solution to the schedule management problem for students.

By taking advantage of the existence of ACS UPB Mobile, we managed to find an appropriate environment for the proposed software solution to be integrated within. As the application already had a timetable implemented, we successfully contributed to it, including more event types to reflect all of the ongoing academic activities better, enhancing the user interface and experience, and adding new functionalities to help the students track and solve tasks faster.

Researching the other applications that implement similar features to the ones proposed, we found general structures that can be very powerful in tracking day-to-day tasks in the hands of a proficient user. As university events can also be translated into these applications, an inconvenience was the initial effort to transform academic assignments into general events and to keep them updated. We identified relevant elements and managed to implement them inside the application, keeping similar user flows so a transition can be done with less adaptation effort.

We analyzed the existing architecture and found ways to integrate new components, starting from the database, where we added a new collection and integrated new documents inside already existing collections. We managed to implement a new planner feature and add it to the application, ensuring it doesn't disrupt other modules. We continued by creating a good user interface. We followed the

UI and UX guidelines established in the application by the nature of its cross-platform philosophy. We kept common elements and flows of actions, as we wanted to create a pleasant and engaging experience rather than reinventing the wheel. By analyzing the red routes and user flows for our planning solution, we managed to build a coherent user interface. Adjusting already used widgets, we also visually integrated our additions.

In the end, we will iterate through the goals proposed in the introduction chapter and present how they were achieved.

One goal when working for the application was enhancing the already implemented schedule components. We managed to tie together the existing class events in the timetable with our implementation of assignments. Therefore, adding, editing, and displaying the new events is now an integrated experience in the application as they behave like the other ones.

Another goal was to improve the user interface and experience. We did that by adding new widgets in crucial places, like the homepage and the classes information page, that show relevant pieces of information about the events. By doing this, we created a better experience for the students that use the application to track their schedule. Also, based on the pre-existing information, we generated relevant graphs to help in visualizing and understanding how activities are distributed in relation to time.

The general goal was to integrate the new components and user flows with the already existing ones in a way that feels natural so that the result brings the application a step closer to being a universal tool for any student. By adding the planning features, we also gave the application a new purpose as a mechanism for tracking tasks. Now, the user has the ability to customize its events by setting his own goals or by hiding them if they are of no interest. Adding a new page for the planner was an efficient way of separating the tasks from the other events, as it encourages the users to start using this feature as a tool to gather relevant information quickly.

While working on the application, we had to follow good coding practices and plan ahead of time. It was a broad experience, as the development required learning Flutter, understanding the already existing architecture and how to adhere to it, user behaviors and how to design the interface to be more friendly.

7.2 Prospect

7.2.1 Additional features

As the module described in this paper lays the foundation of the personalization and automatization of the university schedule, an opportunity for improvement arises. The features implemented can be considered a starting point, as they handle the general problems that the students have to deal with when trying to manage their time. A continuous development process can be established so that our solution evolves with new academic demands.

One example, using the power of Firestore, changing the data model for the assignments is simple, as the document structure is flexible. Changes within the application are also easy to do, as all the methods that deal with data transfer are contained in a single provider. This continues the design philosophy of a general-purpose application that can be easily customized to satisfy the needs of students.

New feature ideas are also taken into account, as the development team behind the application is also part of the target users, and feedback can be easily obtained. By doing regular checks over the ratings and suggestions given by the users, and also directly engaging with them in online communication groups provided in applications like WhatsApp, Facebook, Teams, we can obtain valuable insights on their satisfaction levels, opinions, and suggestions.

As the project for the planner features is visible for anyone on the Github repository of the application, anyone can directly contribute to the evolution process by raising issues, suggestions, and detailed features inside, and also independently developing upgrades for them to be included in the codebase. Examples for future upgrades are integrating more goals for an assignment, generating more relevant statistics and advice on how a task should be handled, a higher level customization by letting the user create personalized aims.

Two new features that can be added are group tracking, an option for teams of students to all track one assignment. This creates an opportunity for collaboration, as it will encourage and guide all the members of the group to better organize, to periodically post updates on their work, and generally, to track the progress of the task. By doing so, we advance from the problem of helping one student with his time management to organizing the shared schedule of groups of students. The

difficulties of having different work habits and targets deserve a comprehensive solution. The mechanism of goals was intentionally planned to be a separated component from the events, as this offers a foundation for implementing this feature.

The other feature is integrating artificial intelligence to analyze the behavior of the users and guide them to choose a style that best suits them. As more and more users start to use the application for all its features, it can become a suitable environment for student behavior analysis. On one hand, the data collected can help predict and recommend typical actions. On the other hand, the data collected can be used to generate statistics and send them to the university to be interpreted.

7.2.2 StundentHub

While the solution of this paper was in a development phase, the team of ACS UPB Mobile participated in the startup accelerator program called Innovation Labs¹. As the application was build from the start to be adaptable to changes in the academic structure of the ACS faculty, an opportunity was found as other faculties from Romania follow a similar structure. Customizing filters to reflect a new organization type, adding a new database, changing the color scheme and other minor changes are easily doable. After, the application is technically ready to be used inside a new faculty.

The features proposed in this solution were also build to be adaptable to a new structure. Our main concern was that the goals set by the users are general enough to be used in other notation systems while also not too generic, so the users don't get confused. We managed to do that by guiding the users through the interface on what every field of the goal represents and what values should be inserted.

As a final conclusion, the module implemented for the purpose of this paper proposed a solution for schedule personalization and automatization for productivity. Being part of the ACS UPB Mobile, we further customized the solution to fit the needs of the target users better.

¹<https://www.innovationlabs.ro/>

Bibliography

- [1] P. Turla and K. Hawkins, *Time Management Made Easy*. Dutton, 1983. [Online]. Available: <https://books.google.ro/books?id=zf4JAQAAMAAJ>
- [2] S. Deb, "Information technology, its impact on society and its future," *Advances in Computing*, vol. 4, no. 1, pp. 25–29, 2014.
- [3] D. L. Butler and M. Sellbom, "Barriers to adopting technology," *Educause Quarterly*, vol. 2, no. 1, pp. 22–28, 2002.
- [4] E. Peper, R. Harvey, I.-M. Lin, and P. Duvvuri, "Increase productivity, decrease procrastination, and increase energy," *Biofeedback*, vol. 42, no. 2, pp. 82–87, 2014.
- [5] N. L. Russo and B. R. Graham, "A first step in developing a web application design methodology: understanding the environment," in *Methodologies for Developing and Managing Emerging Technology Based Information Systems*. Springer, 1999, pp. 24–33.
- [6] B. Traynor, J. Hodson, and G. Wilkes, "Media selection: A method for understanding user choices among popular social media platforms," in *International Conference on HCI in Business, Government, and Organizations*. Springer, 2016, pp. 106–117.
- [7] Hootsuite & We Are Social. (2021) DIGITAL 2021: ROMANIA. [Online]. Available: <https://datareportal.com/reports/digital-2021-romania>
- [8] StatCounter. (2021) Mobile Operating System Market Share Worldwide. Date accessed: 28.06.2021. [Online]. Available: <https://gs.statcounter.com/os-market-share/mobile/worldwide>

- [9] I. Alexandru, "Design and implementation of a cross-platform mobile application that facilitates student collaboration," 2020. [Online]. Available: https://github.com/student-hub/paper/blob/master/Design_and_implementation_of_a_cross_platform_mobile_application_that_facilitates_student_collaboration.pdf
- [10] Universitatea "POLITEHNICA" din București. (2012) REGULAMENT de organizare și funcționare a Centrului de Resurse de Informare și Documentare (CRID) și de recunoaștere a perioadelor de studii efectuate în străinătate. [Online]. Available: <https://upb.ro/wp-content/uploads/2018/02/5REGUL1.pdf>
- [11] Universitatea "POLITEHNICA" din București . (2013) REGULAMENT privind organizarea și desfășurarea procesului de învățământ universitar de licență în Universitatea POLITEHNICA din București. [Online]. Available: https://upb.ro/wp-content/uploads/2017/11/regulament_licenta.pdf
- [12] Universitatea din București. (2019) REGULAMENT privind activitatea profesională a studeților. [Online]. Available: <https://unibuc.ro/wp-content/uploads/2019/10/Regulament-privind-activitatea-profesional%C4%83-a-studentilor-modificat-2019.pdf>
- [13] Academiei de Studii Economice din București. (2020) Carta Academiei de Studii Economice din București. [Online]. Available: https://www.ase.ro/ase_responsive/Metodologii/carta.pdf
- [14] Universitatea Tehnică din Cluj-Napoca. METODOLOGIA DE EXAMINARE (EVALUARE) A STUDENȚILOR DIN UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA. [Online]. Available: https://www.utcluj.ro/media/page_document/157/Metodologia_de_examinare_a_studentilor_Ziuz0K1.pdf
- [15] Carl Sjogreen. (2006) It's about time. Date accessed: 28.06.2021. [Online]. Available: <https://googleblog.blogspot.com/2006/04/its-about-time.html>
- [16] StatCounter. (2021) Search Engine Market Share Worldwide, May 2020 - May 2021. Date accessed: 28.06.2021. [Online]. Available: <https://gs.statcounter.com/search-engine-market-share>

- [17] WebsiteBuilder. (2021) 104 Fascinating Mobile Marketing Statistics & Facts for 2021. Date accessed: 28.06.2021. [Online]. Available: <https://websitebuilder.org/blog/mobile-marketing-statistics/>
- [18] A. Griffin. GOOGLE CALENDAR 'GOALS' UPDATE USES ARTIFICIAL INTELLIGENCE TO MAKE ITS USERS INTO BETTER PEOPLE. Date accessed: 29.06.2021. [Online]. Available: <https://www.independent.co.uk/life-style/gadgets-and-tech/news/google-calendar-goals-update-uses-artificial-intelligence-make-its-users-better-people-a6982516.html>
- [19] A.-R. Adl-Tabatabai, M. Cierniak, G.-Y. Lueh, V. M. Parikh, and J. M. Stichnoth, "Fast, effective code generation in a just-in-time java compiler," in *Proceedings of the ACM SIGPLAN 1998 conference on Programming language design and implementation*, 1998, pp. 280–290.
- [20] N. Kuzmin, K. Ignatiev, and D. Grafov, "Experience of developing a mobile application using flutter," in *Information Science and Applications*. Springer, 2020, pp. 571–575.
- [21] Tadas, Petra. (2020) Introduction to Flutter BLoC. Date accessed: 28.06.2021. [Online]. Available: <https://itnext.io/introduction-to-flutter-bloc-524510218c86>
- [22] IBM Cloud Education. Three-Tier Architecture. Date accessed: 28.06.2021. [Online]. Available: <https://www.ibm.com/cloud/learn/three-tier-architecture>
- [23] Kayfitz, Brian. (2019) Getting Started with the BLoC Pattern. Date accessed: 28.06.2021. [Online]. Available: <https://www.raywenderlich.com/4074597-getting-started-with-the-bloc-pattern>
- [24] E. Müller, "Web technologies on the desktop: an early look at flutter," B.S. thesis, 2021.
- [25] Apple. Human Interface Guidelines. Date accessed: 29.06.2021. [Online]. Available: <https://developer.apple.com/design/human-interface-guidelines/>
- [26] J. M. Wilson, "Gantt charts: A centenary appreciation," *European Journal of Operational Research*, vol. 149, no. 2, pp. 430–437, 2003.

Appendices

Appendix A

Calendar interfaces

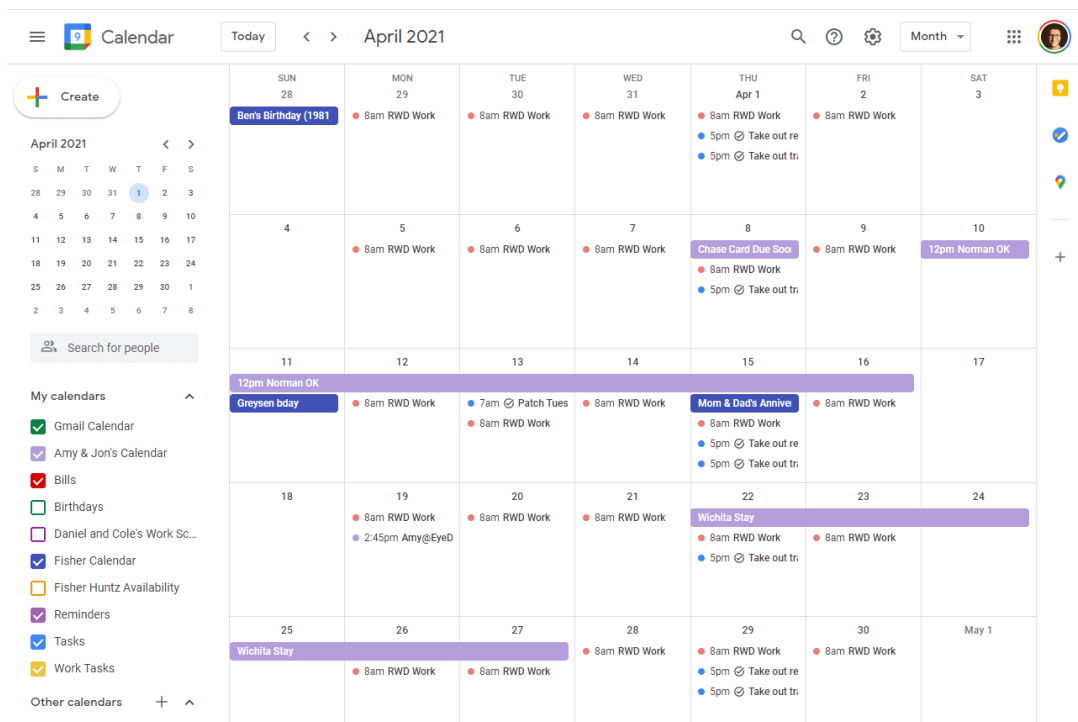


Figure A.1: Google Calendar interface

1

¹<https://www.lifewire.com/google-calendar-review-1357929>

Appendix A. Calendar interfaces

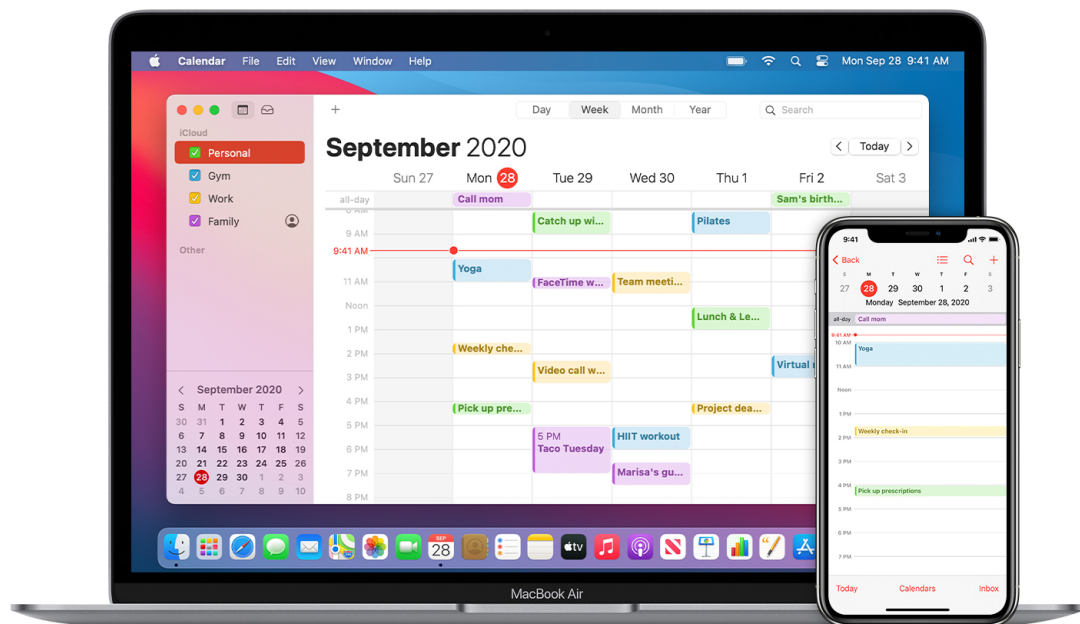


Figure A.2: Apple Calendar interface

2

²<https://support.apple.com/ro-ro/HT202337>

Appendix B

UML diagrams

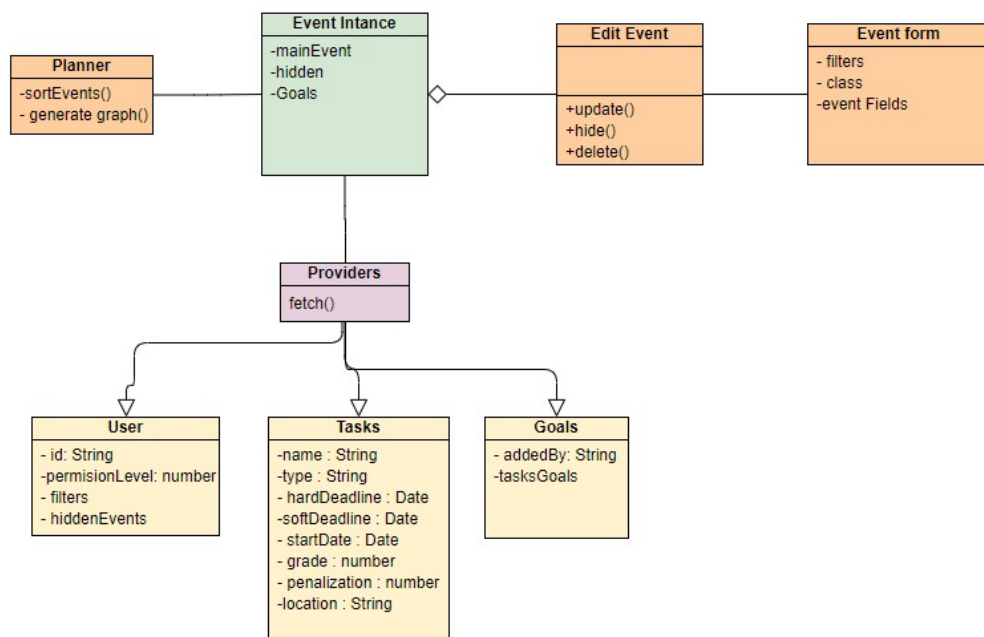


Figure B.1: Class diagram

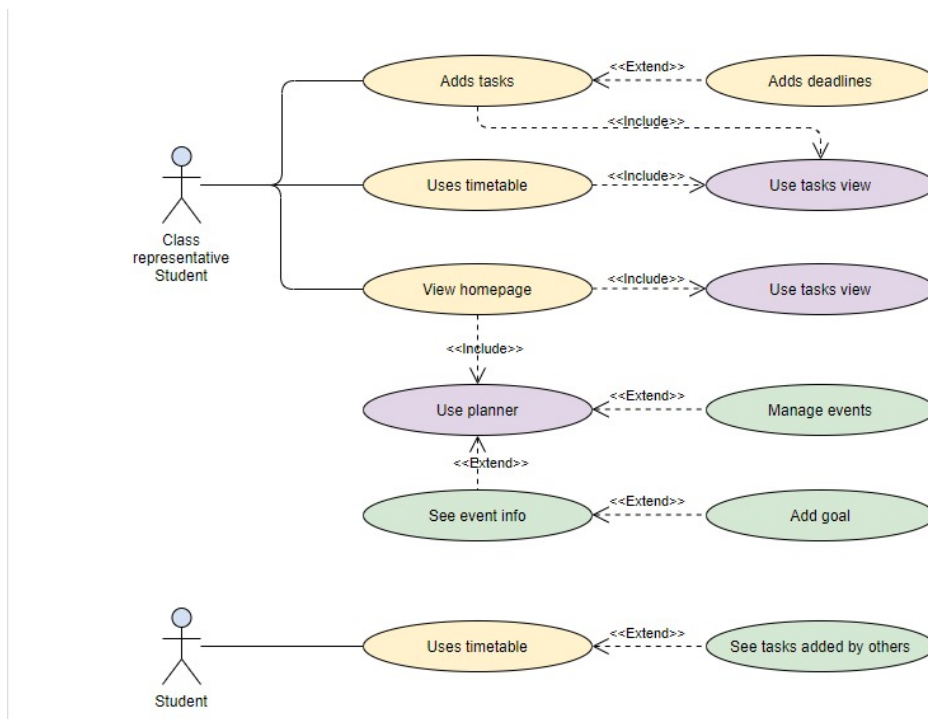


Figure B.2: Use-case diagram