Technical University of Moldova

Faculty of Computers, Informatics and Microelectronics

Software Engineering and Automation Department

The Basics of Software Application Development

Modular Organizer

Authors:
Branzeanu Marinela
Cambur Dumitru
Evstafiev Nicolae
Gavirlita Ion
Nichiforov Maxim

Supervisor:
Popa Dorin

Chișinău, 2020

Contents

1 Abstract

With the development of internet and online services, as well as PaaS (platform as a service) the current lack of organizational and/or time management platforms is still not filled due to the intentions of big tech leading companies trying to outperform each other by trying to lock in the user in their ecosystem. The following report aims to analyze and give a better insight on the way a comfortable and powerful management platform should look like.

2 Introduction

2.1 Domain analysis

2.1.1 Abstract

Domain analysis is the process by which a software engineer learns background information. Our team had to learn sufficient information so as to be able to understand the problem and make good decisions during requirements analysis and other stages of the software engineering process. The word 'domain' in our case means the general field of business or technology in which the customers expect to be using the software.

Some domains might be very broad, such as 'airline reservations', 'medical diagnosis', and 'financial analysis'. Others are narrower, such as 'the manufacturing of paint' or 'scheduling meetings'. People who work in a domain and who have a deep knowledge of it (or part of it), are called domain experts. Many of these people may become customers or users.

In order to best understand our domain and develop a wider understanding of the task overall our team has done an extended research regarding the topic at hand.

In order to perform a good analysis as already mentioned, we gathered information from whatever sources of information were available: including domain experts; any books about the domain; any existing software and its documentation, and any other documents we could find. The interviewing, brainstorming and use case analysis techniques discussed later in this chapter can be of assistance with the domain analysis.

As a software engineer, you are not expected to become an expert in the domain; nevertheless, domain analysis can involve considerable work. Benefits such as Faster development, Better system understanding and an Anticipation of extension will be of great help for future development and make the work worthwhile.

2.2 Problem formulation

Digital calendars are a mess. They're a crucial part of modern life, especially as remote work becomes more prominent. They help employees/students to organize their day and bosses/managers can actually make sense of the work that's getting done. However too often, most of this work is just about ... dealing with calendars. Employees/students fill all their half-hour boxes with tasks, meetings and personal commitments, only to have bosses, clients, studies and co-workers/colleagues steal that time, one unexpected invite at a time. Once-focused days turn into a haphazard series of too-long meetings and too-short breaks, with little time to get actual work done.

Therefore, for our project we decided to tackle the problem of the big quantity of the calendars, reminders, task apps, in day-by-day life. This is a big problem that we encounter and this is what we base on.

From our experience, we regularly get tasks which contains deadlines. The problem is that we get them on different platforms from several organizations like university, work, study centers or other places. We get many daily events which take place on separate platforms. We can take and rewrite and introduce the same information multiple times on different calendars and this will

take a lot of time and efforts. And all this can increase the chance of error occurrence, much less people that are working with many clients and customers which can daily organize different events on different platforms. They deal with focus loss and this can influence their and other lives like a forgotten meeting, undone work or missed deadline.

2.3 Solution concept

An online platform designed for vast groups of users for day-by-day life activities to more specific workflows and study schedules. The platform aims to have a calendar with a list of tasks and a lot of modules that can be chosen and integrated by choice as desired by anyone. Users will have the option to connect from a catalog of helpful modules which will then be integrated into the main calendar app for a better and more personalized user experience. Like a FAQ tab or a 'get-started' option, we will plan to introduce a platform-wide smart-assistant that will help the user step-by-step how to utilize the platform. The end goal is a platform that will help people to organize time, as combining many platforms in one will improve work speed and will help with reminders which will make planning effortless and a lot more enjoyable than using the traditional organizational application stacks.

Also, and also our second aim of this platform is to have all schedules arranged on one screen where users can easily interact with them as well due to the introduced assistant, interaction with the platform will be more interactive and intuitive. We want users to be able to manage their time-efficient using our platform, due to installed custom modules there are no limits on the site's abilities in terms of interaction with users, it can show incoming tasks, show the traffic situation, shows the bus in the traffic, talk about weather conditions and recommendation about the weather. Also, it can communicate with other sites via API (Application Programming Interface) (for syncing purposes) to increase even more the possibilities of user managing time using our platform like booking a place to a café, make invitations for events, booking seats to the movie theatre.

Furthermore, to motivate users to use all possibilities of the platform as well as experiment with new features we will introduce a challenges/ranking system based on how much they use all features of the web app.

Not to get too messy at the start assistant will companion users through all necessary steps to understand how to integrate and use modules as well as how to connect and manage new calendars and how to use them all the potential of the platform.

2.4 Motivation

General Motivation

In general, this calendar will help to arrange and scheduling the tasks meeting and all daily routine so users can see how they spent time and how much time they have to do a type of work, included assistant will help to choose what tasks to do next, just introducing all in form of a list of tasks, daily routines and meetings all with time intervals, and the calendar will fill the gaps in calendars confirm the introduced list, in time of creating the list the can be determined a bunch of characteristics of this task its duration, time buffer for getting to a place and so on. Besides, if two

events are colliding with each other the calendars will ask these two events to move to another time or simply delete it.

Personal Motivation

As a student, we have to plan study schedules where we can concentrate on studying so because we get enormous amounts of homework to do, we have to plan and schedule all work hours so we can make all the work done but because we do not get all the task in one time we have to plan and be attentive with time scheduling on all sorts of things. For instance, in one day, we can get a bunch of tasks from university or from work we can get nothing this irregularity of getting tasks can get complicated to schedule moreover if it is something new, we can't exactly say how much time this will cost to do so, we have to be very flexible with switching tasks and to be present at all seminars and laboratories. As students, we want to make a platform that will make a flexible list where we can fast and easily schedule our next day-week of work.

3 Domain Analysis

3.1 Impact of the problem over the domain of study

Is there anything more symbolic of the absurdity of modern life than how hard it is to make plans with friends?

First, you must decide how to get in touch. Email? It's fine, but then you have to deal with 20 replies. Apps like WhatsApp or Signal rarely work because not everyone is on them. And relying only on Facebook guarantees you miss those friends who no longer — or never did — have an account. In short, it's a mess before you even get started.

Then there is the almost comical back and forth of trying to find a time and day that works for everyone in our busy, scattered lives. Meetings, soccer practices, vacations, tickets to see a show, babysitting dilemmas — they are all part of an endless litany of stuff we fill our days with. It's a wonder we see our friends at all.

Sure, it's great fodder for standup comedians, but it's also sort of maddening: As digital tech has made many basic things like communicating, shopping, or ordering food more convenient, getting some pals together for drinks is apparently beyond us.

The problem? The calendar app is broken and sorely in need of reinvention.

ADVERTISEMENT

Consider how the calendar compares to email. Though they both started as "professional" tools, now essentially everyone who has access to the internet has an email account. The same cannot be said for calendars. That is an indictment of modern calendar apps all on its own. Sure, some of your nerdy friends send you a Google Calendar invite, but most wouldn't think to.

To explain why calendar apps are so bad, it helps to think about their roots. While scheduling applications were included with Windows and on the Macintosh from early on, they didn't really take off until the arrival of the internet. In the corporate world, it was Microsoft Exchange and Outlook that came to dominate, and in many cases, still do; even now, 70 percent of Fortune 500 companies rely on Microsoft software.

But those corporate roots are also why calendar apps are so clunky. For people on the same system, features gradually expanded to include things like setting up meetings, showing yourself as busy, and more. Yet, it also meant that while anyone with an email address could send a message to anyone else with one, calendars remained cordoned off. Now, despite efforts to develop universal standards for calendars, nothing has really taken hold. There are at least three or four choices, instead of just a single universal one.

Yes, fine, this all sounds a bit arcane, and it is. But that's just the problem. Regular people need calendars, too. Imagine how much easier the basic act of getting people together or organizing your family's time would be if everyone could simply propose times for an event and have other people sign on or not, or have a fast, clean, shared calendar that everyone can see and edit quickly. Right now, because there is no universal standard, that is either impossible or needlessly difficult.

What needs to change? First, the digital calendar needs to become more like email: Everyone should be able to have one, and regardless of what app you use, all the basic functions — you know,

proposing or listing a place, blocking off time to prevent overlap, invite lists, and so on — should all be standard. It shouldn't matter if you were on a Mac or on Microsoft's Outlook or in Google Calendar. It should just work.

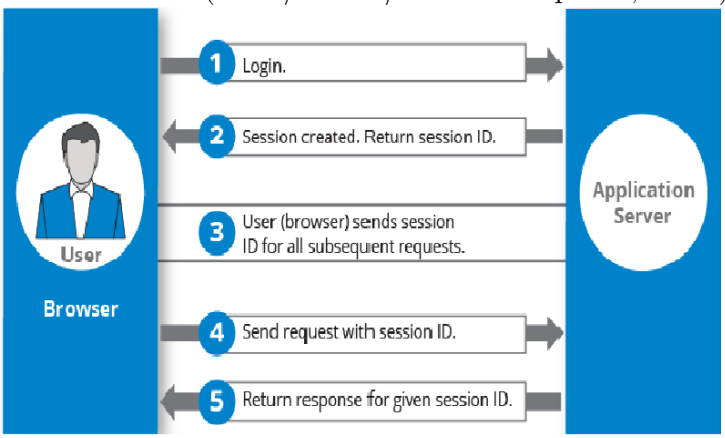
To talk of a calendar app as the solution to our social woes is perhaps to miss the bigger picture: If our lives weren't both so regimented and segmented, perhaps an app to carve out one's time into chunks might not be necessary. And that's to say nothing of the competitive market forces that would get in the way of making the calendar better: Facebook wants its calendar not to work with others so you stay on its platform, and Microsoft, Google, and others are no different.

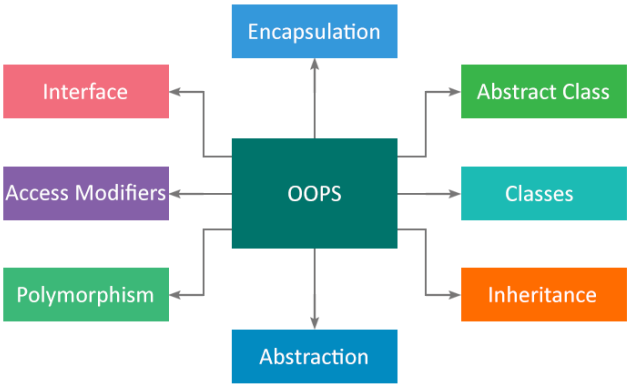
But it's for this reason why it's important to push for a reinvention of the calendar app. The web and digital tools are at their most useful and equitable when they are predicated on standards — when they allow whoever wants to use them to do so, without restriction, and without the coercion of making everyone use the same tool.

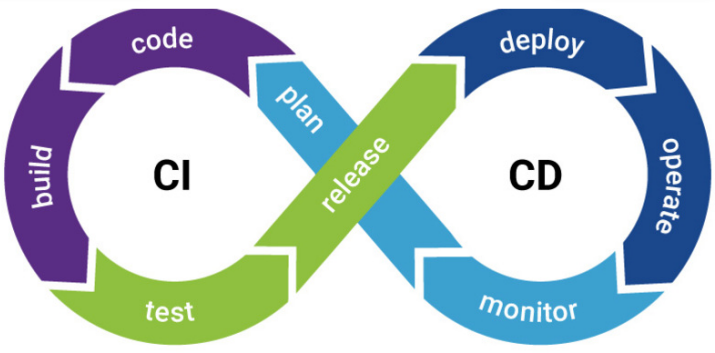
We decided to go with this project idea due to the sheer amount of learning possibilities it offers – the main requirement would be a better understanding of OOP/PO/CRUD concepts, as they will be main way of building our codebase for the project.

The usage of sessions and relational data-bases for storing session and user related information which has a steep learning curve but is well worth the investment as it is a fundamental practice in building todays web services and application.

And finally, the backend and technology stack used for delivering the project – this requires a good understanding of CI/CD practices as well as different frameworks and a good knowledge of the server-side part and how things work over the network (GET/POST/HTTPS requests, etc...)

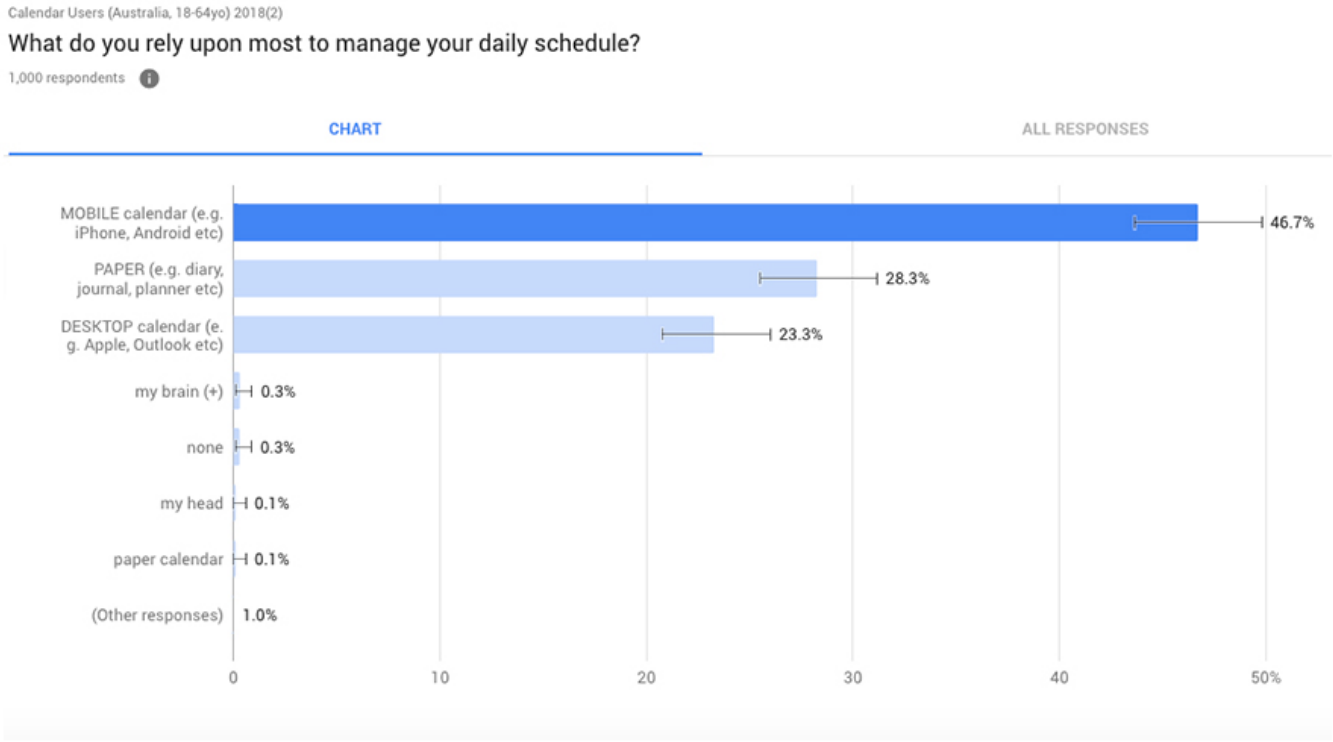




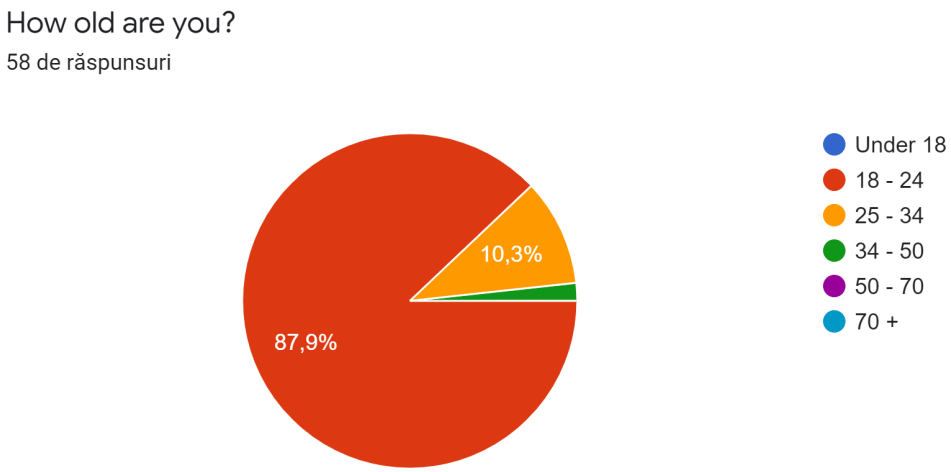


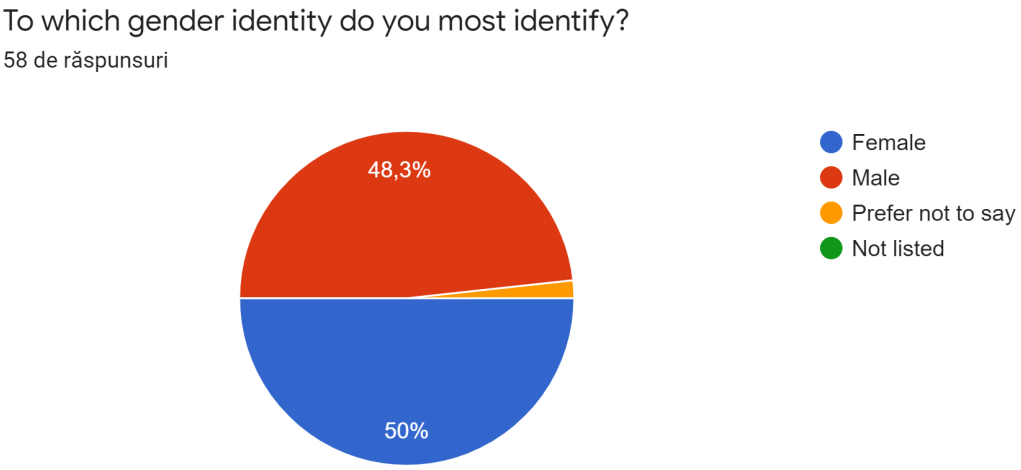
3.2 Target group

In order to start working we check the amount of people that are using digital calendars which we took results from ECAL annual survey aimed on aged between 18-64 years from all geographical locations across Australia. The results are that 70% rely on digital calendars.

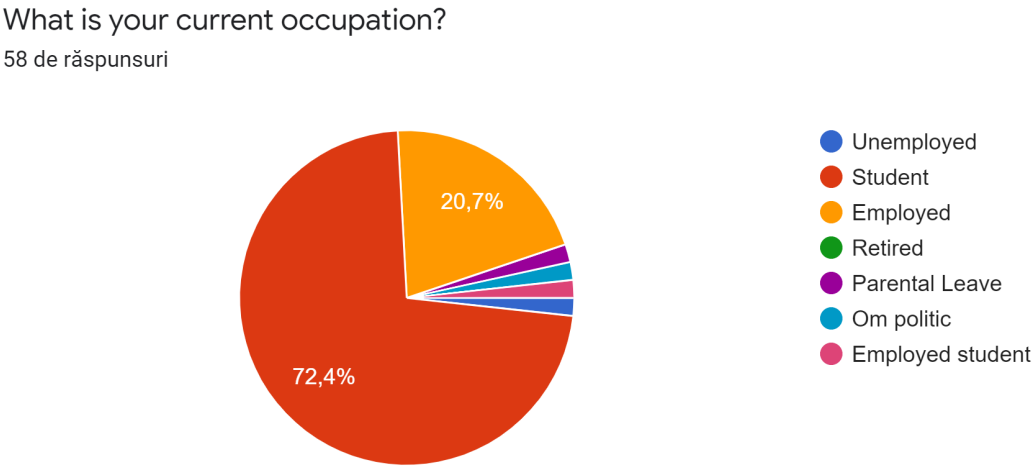


To understand the target group and customer interest in such a product, we conducted our own survey in which 58 persons took part. This survey helped us to understand to which age group, gender and occupation future customers will identify.





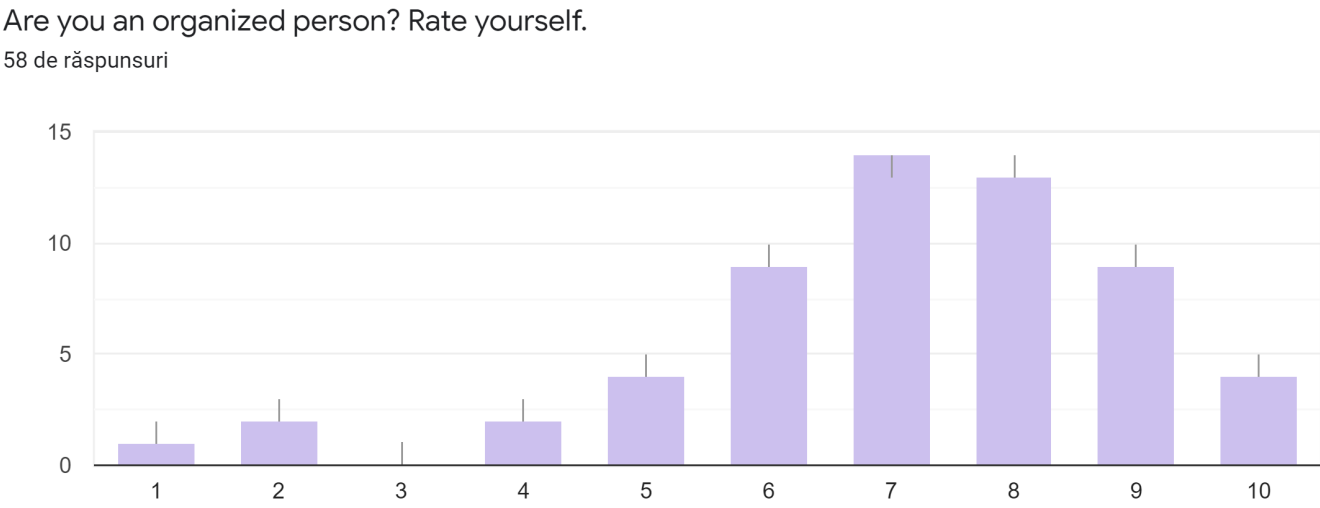
The main advantage of this survey that gender majorities are in equal parts and we have a fair opinion about our product.



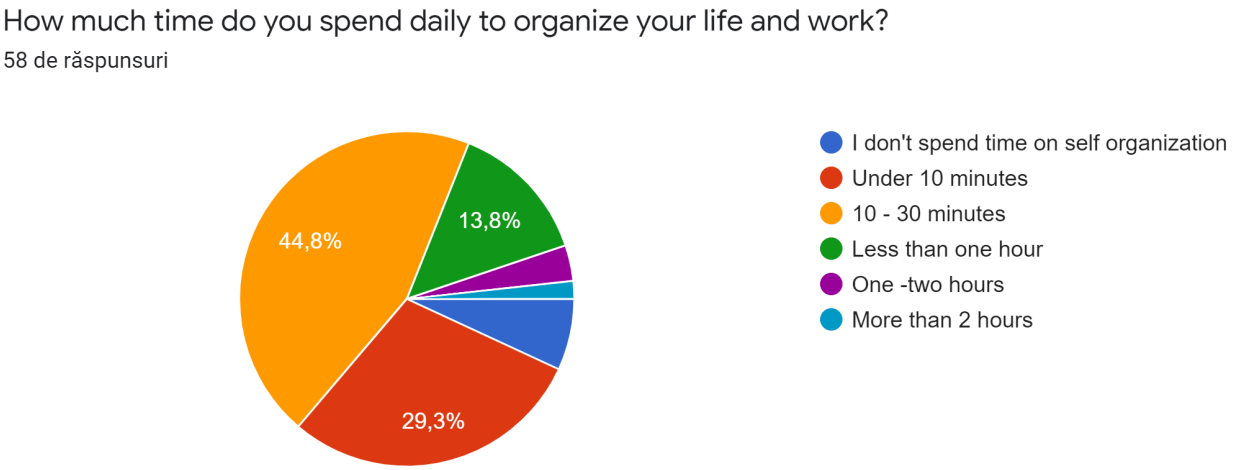
Here we can see that the main parts of audience will be students and employed people. At the end of survey, we had the input line for suggestions, ideas and someone that took part in survey had written to add one more field with employed student. We thought that better will be to add 'others' field such that people will have more freedom to describe their current occupation. This way, we have politicians here. After these 3 questions we understood that the leading audience of the app will be English speaking active study-working people aged between 16 and 35 which have a lot of tasks events which are related to teams or companies that work on different platforms, or for startuppers who have an extremely flexible schedule and need a platform which will help to organize all tasks and events on one board.

3.3 Customer validation

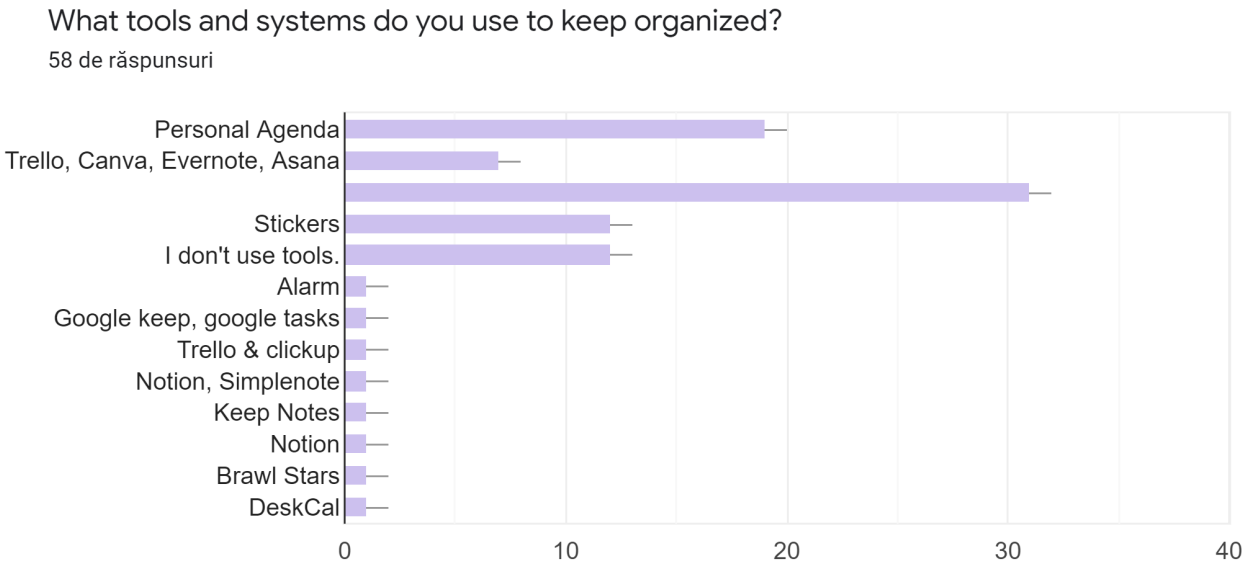
Another part of survey had the aim to find out if people are interested in such an app or not. Firstly, we had to find out which are their habits about organization and time management.



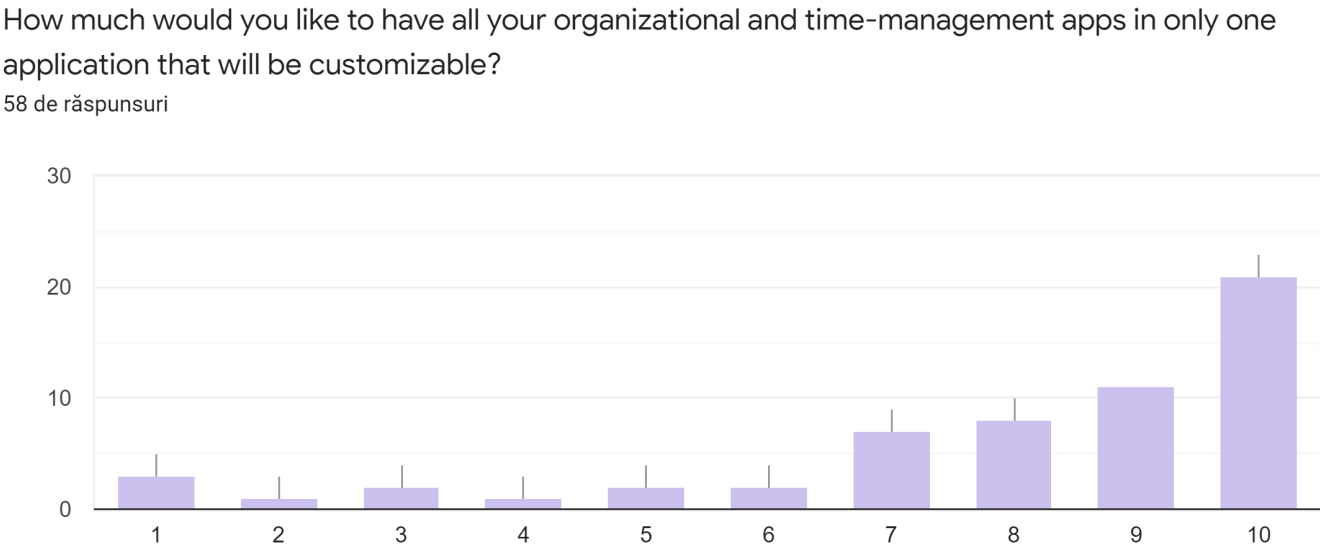
As we can see most people consider themselves moderately organized, because more than 25 people grade themselves with and 7 or 8. So we understand that people are not enough satisfied of their coordination.



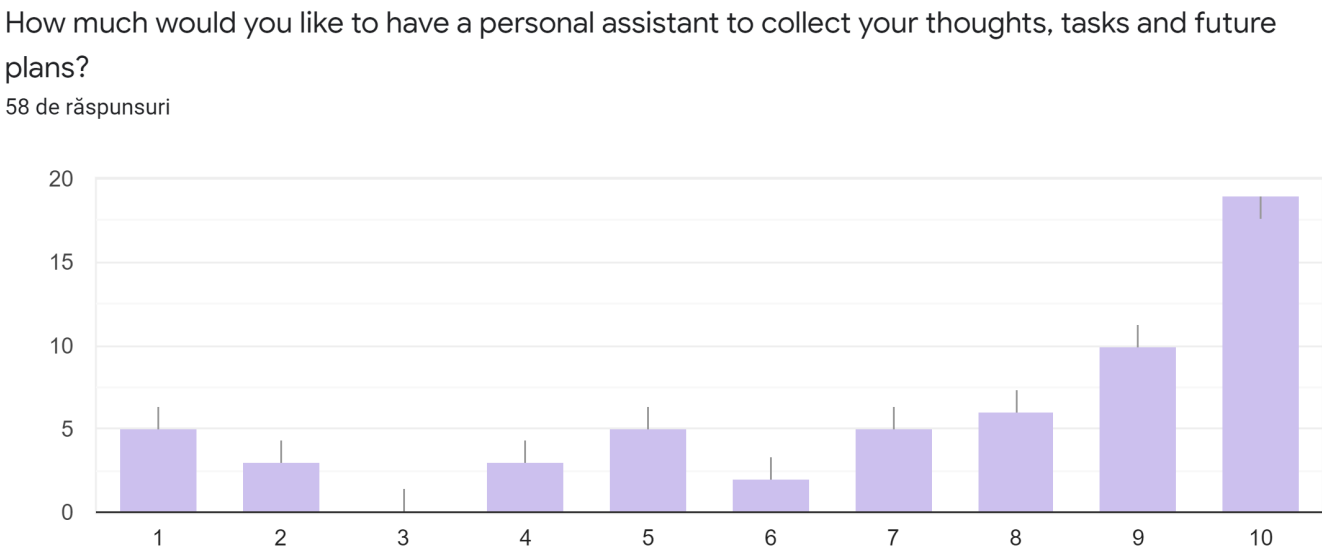
44,8 % of people spend 10-30 minutes per day to plan their life and work. Per year they spend 7300 minutes on this. It is 121 hours, 5 days. In their entire life they spend 250 days on arranging. There is a lot.



31 humans use Google Calendar, Outlook Calendar, Todoist. This means that people like to view their task, activities, meetings in a calendar way of displaying. The second place take agendas with 19 votes. This means that even that we live in very digitalized world people choose to use the paper-based type of managing their activities and thoughts.



30 assume that their desire of having an organizational and time-management app that will be customizable and will be integrated with other calendar apps is graded with 9 or 10. This is more than a half of total number of people took part and this means that the interest in such an application is highly shown. This means that developing such a concept has the right to be in production.



29 people shown surely that they want to have a personal assistant to collect their thoughts, tasks and plans. It can be understood that future customers are highly interested in both of our main advantages over competitors.

3.4 Competition

Figure 3.1– Google Calendar Logo

a) Google calendar is not the most intuitive app of using tasks and schedules it has a lot of special conditions to get the desired result also it is consisting in a big platform google where cam communicate with other google application so it easy to work

We have made a research and found out the common problems of Google Calendar:

If the calendar doesn't belong to you and hasn't been shared with you, Google will not provide the Summary or Description fields. The solution is simple, have the owner share the calendar with you (even if it is public)!

Repeated events in Google Calendar aren't supported with the "New Event" trigger in Zapier. A repeated event would cause an infinite amount of Tasks to occur since repeated events last forever. However, repeated events can be used with the "Event Starts" trigger.

All-day events in Google Calendar end at midnight on the last day, so they're exclusive of the end date. For example, an event created via Zapier for August 10 - August 15 will appear to span August 10 - August 14 in the calendar UI, because the event will end at 12:00:00 on August 15.

To fix this, you can either update your trigger data so that it lasts for an extra day, or you can modify the end date in your Zap directly by adjusting the date/time to include +1d. This will cause the event to last for an extra day.

If you're using the "Quick Add Event" action instead of the "Create Detailed Event" action, there are some specific guidelines you'll want to follow to make sure that Google Calendar can interpret the date and time correctly.

If you set the start date time and end date time to the exact same time it's possible that you might not see the event in Google Calendar when you are looking at the calendar in some views. If you click on agenda view in Google Calendar you will see the event show up. Google Calendar doesn't show events with a length on 0 minutes in some of their other views.

Unfortunately, Google changes the ID of events on their auto-generated calendar in a way that causes the same event to trigger multiple times. We don't recommend building zaps on Google generated calendars because of this.

Google is finicky in recognizing dates/times when using Quick Add, and you'll have to include the information in a format that Google can recognize. Specifically,:

Event details need to follow the order: what, where, when. Events created with the details in a different order may not be parsed correctly.

If you enter an event with no date, it will be added at the next available time in the future

If you don't enter a start time, an all-day event will show up

If you don't enter an end time, an hour-long event will show up

Unfortunately the Quick Add Event action doesn't allow you to invite users to an event, it'll just add it to their calendar. If you'd like to send the user an invitation that they can accept/reject, you'll want to use the Create Detailed Event action instead.

Figure 3.2– SuperSaaS Logo

b) SuperSaaS is good at creating group schedule, but because of the design of the site is hard to understand what happens, saves the situation the support and tutorials page, it supports the work with forms directly when accessing the schedule which is very efficient in scope to get

additional information but it faces issues when it comes to sharing and working across multiple time zones

Figure 3.3– Outlook Logo

c) Outlook to get all the possibilities of the calendar you need to pay, is free for students and have many integrated applications which can communicate direct whit each other that increase productivity it is a combination of calendar and email it is very easy to share about events and set tasks and reminders for groups or selected people but because outlook is a system of big amount of interconnected tools is very hard to understand at start also there are problems with meeting approving, storage security and automatic secure control of account.

Figure 3.4– Jorte Calendar Logo

d) Jorte Calendar theme changing, can use free but to get the chance to change the appearance of the calendar you need to pay subscription but it faces problems with synchronization between other calendars and problems with time zones calibration.

Figure 3.5– Calendly Logo

e) Calendly it is app which works very good with establishing new meetings and events because it gave you to choose the rules that all participants can choose between the permission you establish it is time zone but the free users are limited to 1 calendar and basic functions which do not let user to feel the power of this app also even if is paid version it is facing a lot of

problems with reminding the meetings and problem with booking system and there is no proper customer service.

Figure 3.6– Clockwise Logo

f) Clockwise Protecting travel time, Automatic color code, personal and work calendars in sync, resolving meeting conflicts, handling different time zones, and always saving time for lunch.

Figure 3.7– Reclaim AI Logo

g) Reclaimai, flexible scheduling finds and defends time on calendar, makes and schedule automatic regularly routine that can be chosen from a customizable list

4 System Design

4.1 Objectives

The system design objectives are:

- Establish the final expected result as idea/sketch
- Sketch the functionality scheme for each module needed in detail.
- Setup a server with the necessary dependencies.
- Setup a database used for storing user records
- Site mockup for: home page, plugin page, calendar page, login page, register page, account page
- Implement basic interactions between user and platform: registration, login, CRUD (create-read-update-delete) for records, implement calendar feature, import Google Calendar, add plugins
- Implement Companion feature.
- Create the possibility to add user-made plugins
- Live testing and deployment.

4.2 Requirements specification

Requirements to this platform are:

a) Intuitive and easy to use design

b) Assistant which will companion the user all the time

c) FAQ page

d) User Friendly

e) Adaptive web design

f) Deep personalization

g) Seamlessly API integration.

h) Community custom made plugins.

4.3 Full system design description.

Our solution consists of a variety of modules and components. As we adapted client-server architecture, we can divide the application in two parts, first client, which is visible by user, and strictly speaking is a gateway between user interaction with the server-side. Next one is server and it can be seen as the head of our app where most functionality is implemented. From user point of view, he/she enters the site (client), which sends requests to the server to fetch/insert some data from/into the database, do calculations etc. For instance, in Modular Organizer client is represented by front-end written in React JS (mostly on JSX) user can see it, interact and request something from the server. On the other hand, server consists of more than only back end. In summary, our back-end uses Python flask framework, Gunicorn application server, Nginx and MySQL database.

4.4 System components description and motivation why selected system design fits the best solution for team selected problem.

To realize our goals, we decided to create next components:

a) Calendar API

b) Plugin API

c) Weather API

d) Todo API

e) Configuration Component

f) Authentication/Authorization Component

g) User, Calendar, Plugin, Weather and Todo Entities.

Firstly, we created three different API's to get the highest flexibility and scalability from our application. Thanks to this design, we can easily distribute this project to team members and even better, any errors related to Calendar API side won't get messed up with Weather API or Todo API. It results in more structured code, esthetically more beautiful endpoints and of course, it is easier to create updates, because if Todo API needs update, Weather API won't need it.

Configuration Component is a must for development control. Every application needs to set up environment variables and e.t.c., and for development and production they will differ. This is why there is a need in this specific component.

Authentication/Authorization Component is created to handle authorization and authentication process which will be used on other components/API's later.

Entities are database components. By following database normalization principles, we created different entities which correspond to the API working with them.

4.5 System components interactions. Description of how system components interact with each other, which technologies and protocols are used and which data payload and data format is used for system intercommunication.

Modular Organizer can be divided into client and server parts. Client consists of html/css, javascript (React JS) and server is represented by python, gunicorn, nginx and mysql. Backend is further divided into plugin, calendar, weather and todo API's. All of this is working under REST architectural style, where data transmission is realized via JSON and HTTP protocols. As it is known, HTTP has a pretty wide range of methods, however our project mostly depends on GET, POST, PUT and DELETE methods. The biggest difference between these methods is their purpose and existence of payload in POST. So, strictly speaking our user interacts via client (front end) with server API's, which uses Nginx and Gunicorn to work with HTTP and WSGI, where last of these is needed for python. For example, user wants to see his/her calendar records for this month on the website. Client makes GET request to the server endpoint under calendar API. As most calendars use ISO8601 standard for datetime representation (like YYYY-MM-DDTHH:MM:SSZ) and Odata V4 protocol to work with data our project has it implemented in our endpoints too. Then server gets this request, it seeks for the time boundaries of calendar records from query string. After, our server connects to the database, finds all suitable fields, packs it into JSON and creates a response with a status code 200 (OK), required headers and JSON in it. As our front end and back end is placed on two different subdomains, we have to additionally allow CORS. It is pretty important to mention that our project has fully working Authentication system which uses JWT (JSON Web Token) to create access and refresh httponly cookies. Also, it is well-known that cookies are vulnerable to CSRF attacks. So, we decided to implement CSRF Double Submit Cookie Pattern and we will also implement 2FA (Two-Factor Authentication) later on. Now let's speak more in-depth about techniques and technologies we used in our project.

HTTP - Stands for "Hypertext Transfer Protocol." HTTP is the protocol used to transfer data over the web. It is part of the Internet protocol suite and defines commands and services used for transmitting webpage data.

HTTP - Stands for "Hypertext Transfer Protocol." HTTP is the protocol used to transfer data over the web. It is part of the Internet protocol suite and defines commands and services used for transmitting webpage data.

Some common HTTP status codes include:

a) 200 - successful request (the webpage exists)

b) 301 - moved permanently (often forwarded to a new URL)

c) 401 - unauthorized request (authorization required)

d) 403 - forbidden (access is not allowed to the page or directory)

e) 500 - internal server error (often caused by an incorrect server configuration)

f) 418 – I am a teapot. (client error response code indicates that the server refuses to brew coffee because it is, permanently, a teapot)

HTTP also defines commands such as GET and POST, which are used to handle form submissions on websites. The CONNECT command is used to facilitate a secure connection that is encrypted using SSL. Encrypted HTTP connections take place over HTTPS, an extension of HTTP designed for secure data transmissions.

NOTE: URLs that begin with "http://" are accessed over the standard hypertext transfer protocol and use port 80 by default. URLs that start with "https://" are accessed over a secure HTTPS connection and often use port 443.

JSON - an open standard file format, and data interchange format, that uses human-readable text to store and transmit data objects consisting of attribute–value pairs and array data types. It is a very common data format, with a diverse range of applications, such as serving as a replacement for XML in AJAX systems. REST - Representational state transfer is a software architectural style that defines a set of constraints to be used for creating Web services. Web services that conform to the REST architectural style, called RESTful Web services, provide interoperability between computer systems on the internet.

Odata (Open Data Protocol) - is an OASIS standard that defines the best practice for building and consuming RESTful APIs. OData helps you focus on your business logic while building RESTful APIs without having to worry about the approaches to define request and response headers, status codes, HTTP methods, URL conventions, media types, payload formats and query options etc. OData also guides you about tracking changes, defining functions/actions for reusable procedures and sending asynchronous/batch requests etc. Additionally, OData provides facility for extension to fulfil any custom needs of your RESTful APIs.

OData RESTful APIs are easy to consume. The OData metadata, a machine-readable description of the data model of the APIs, enables the creation of powerful generic client proxies and tools. Some of them can help you interact with OData even without knowing anything about the protocol. The following 6 steps demonstrate 6 interesting scenarios of OData consumption across different programming platforms. But if you are a non-developer and would like to simply play with OData, XOData is the best start for you.

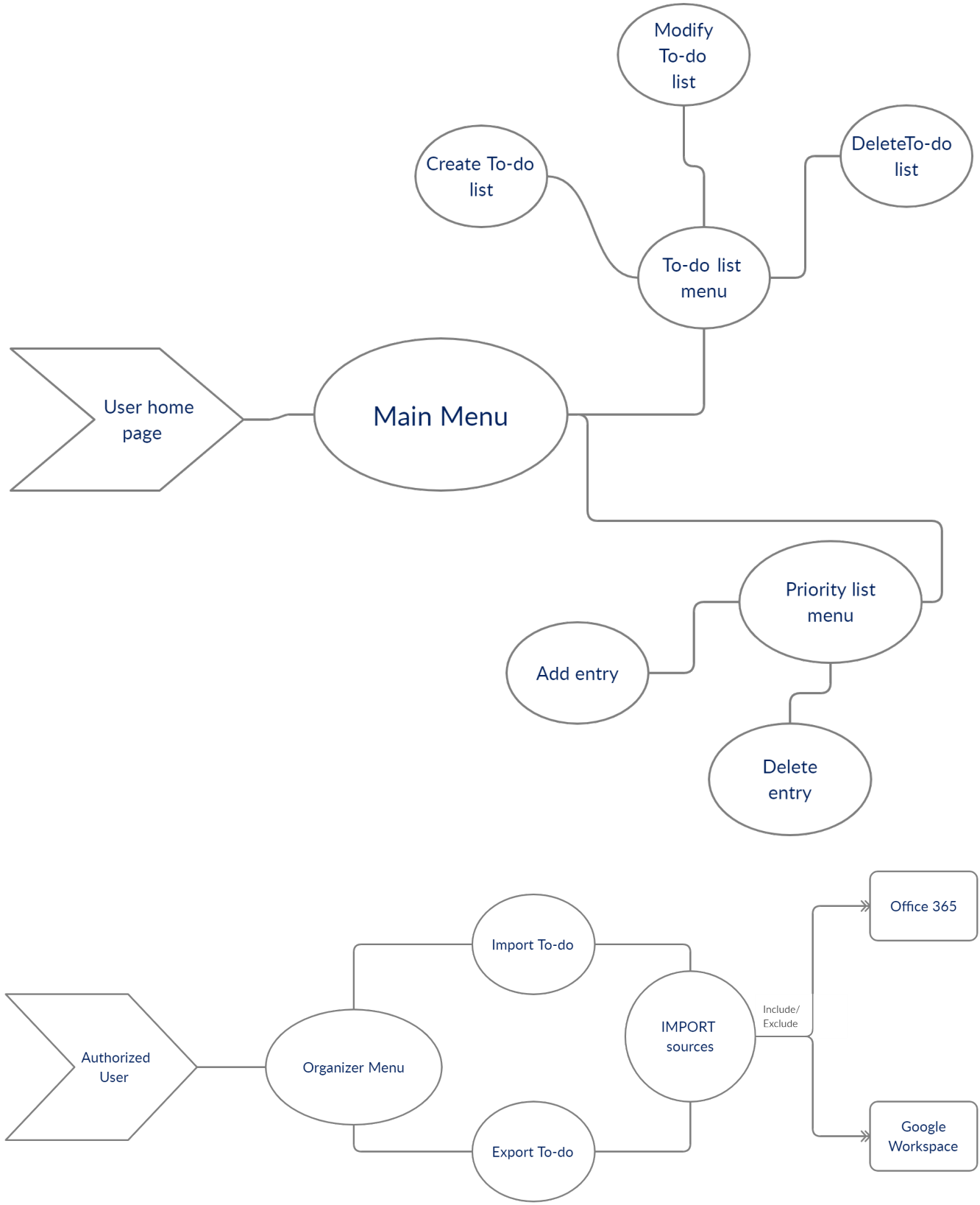
ISO8601 - is an international standard covering the exchange of date- and time-related data. It was issued by the International Organization for Standardization (ISO) and was first published in 1988. The purpose of this standard is to provide an unambiguous and well-defined method of representing dates and times, so as to avoid misinterpretation of numeric representations of dates and times, particularly when data is transferred between countries with different conventions for writing numeric dates and times.

WSGI - stands for "Web Server Gateway Interface". It is used to forward requests from a web server (such as Apache or NGINX) to a backend Python web application or framework. From there, responses are then passed back to the webserver to reply to the requestor.

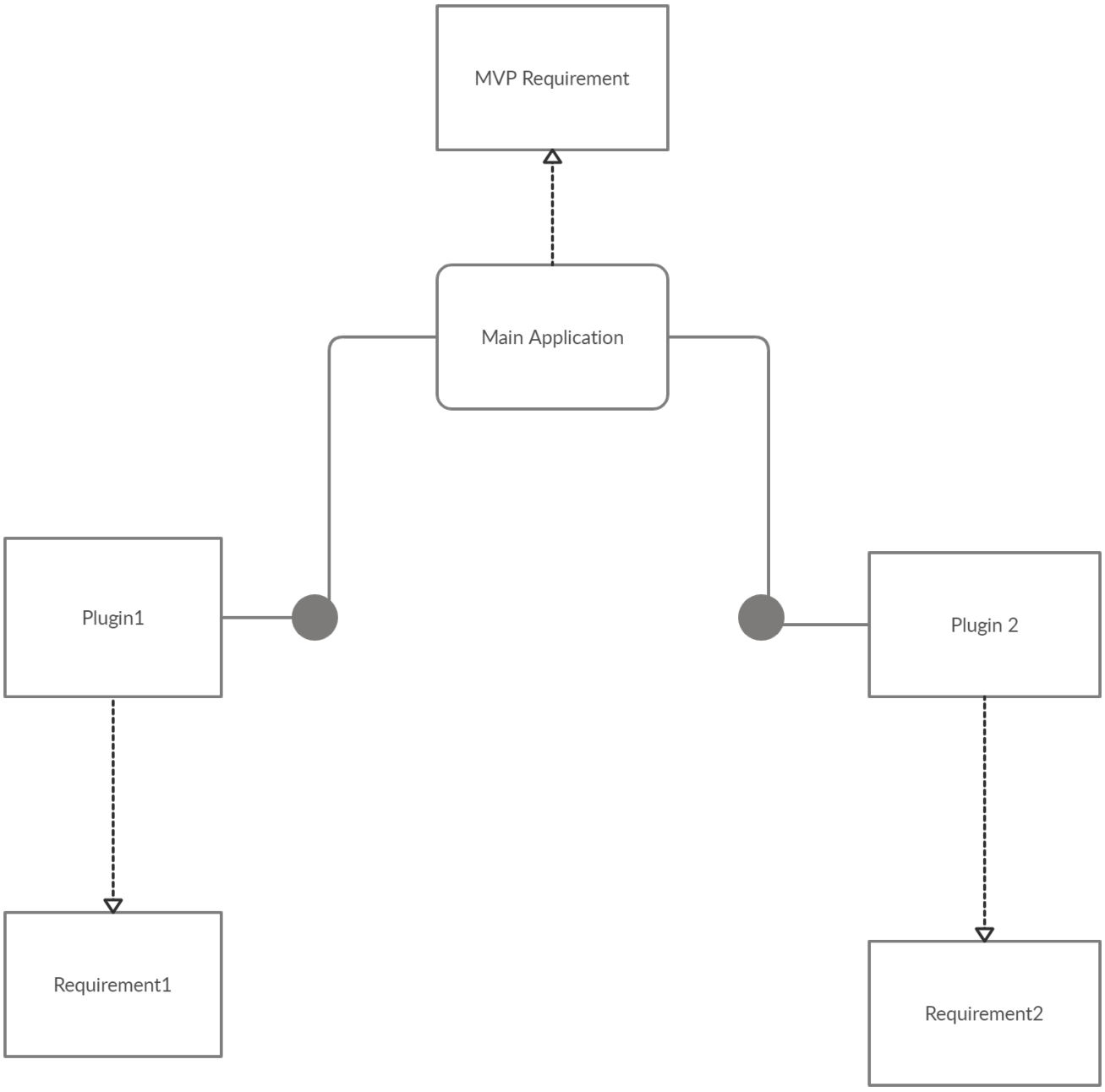
CORS - Cross-origin resource sharing is a mechanism that allows restricted resources on a web page to be requested from another domain outside the domain from which the first resource was served. A web page may freely embed cross-origin images, stylesheets, scripts, iframes, and videos.

4.6 UML Diagrams

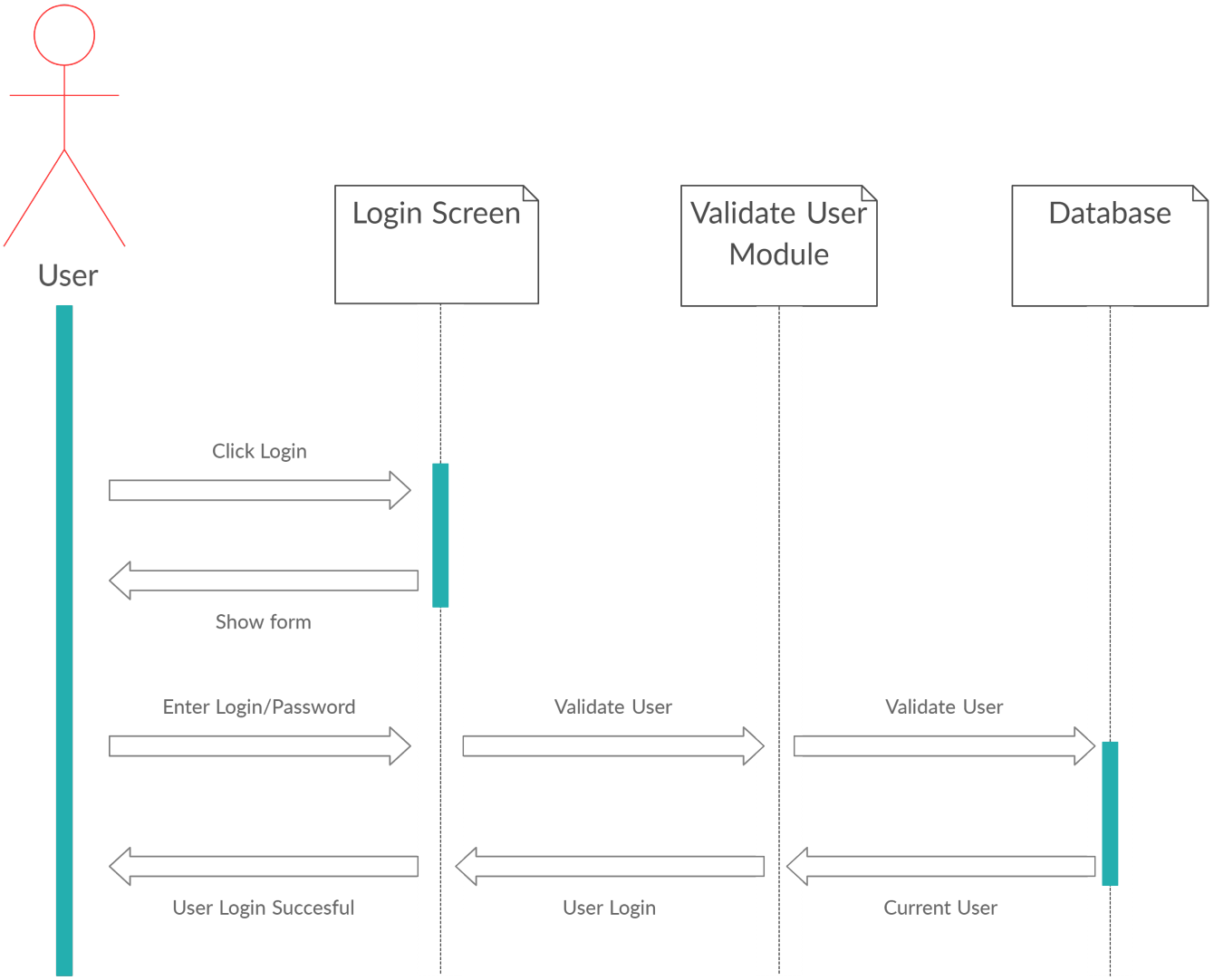
4.6.1 Use case

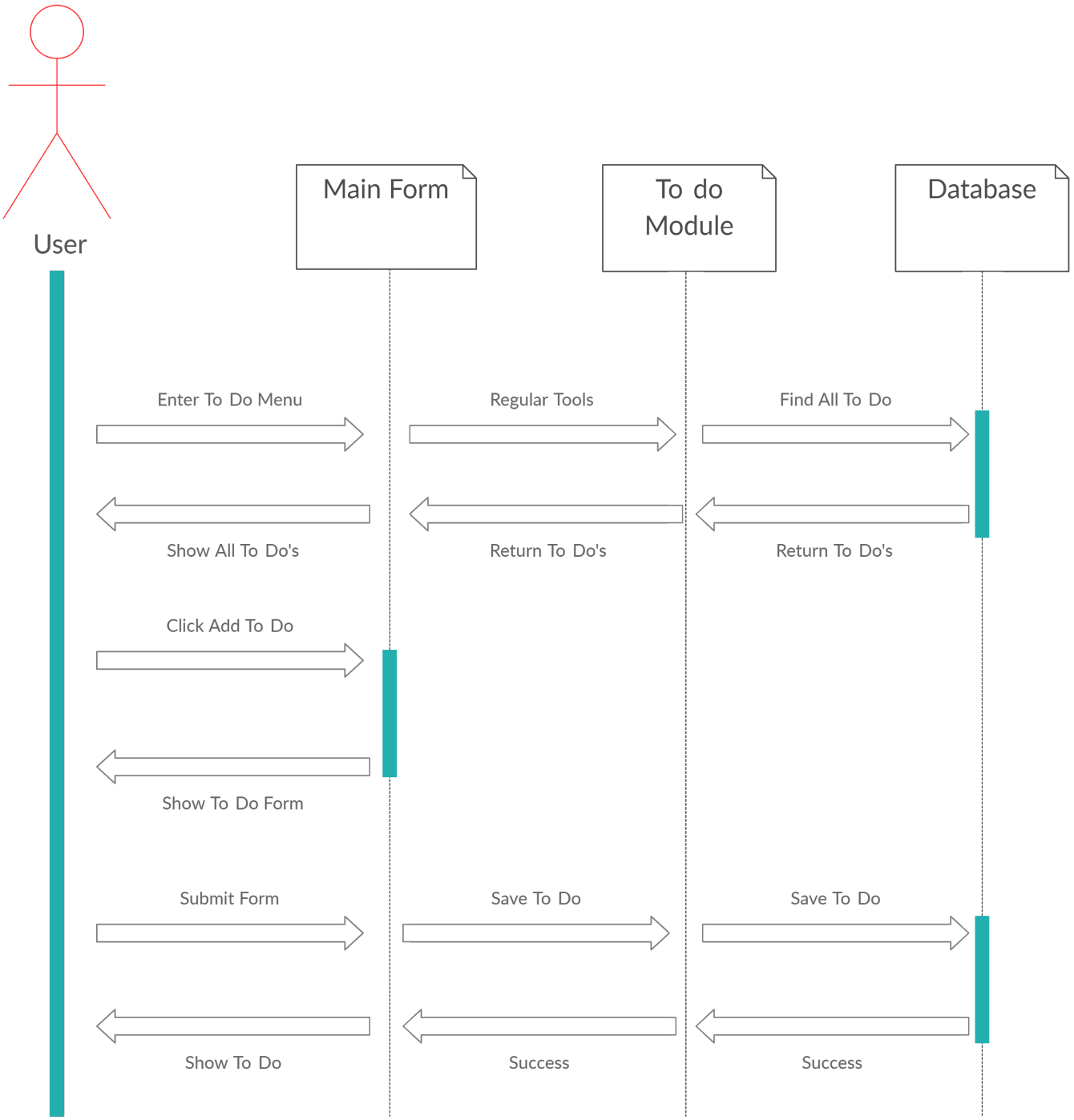


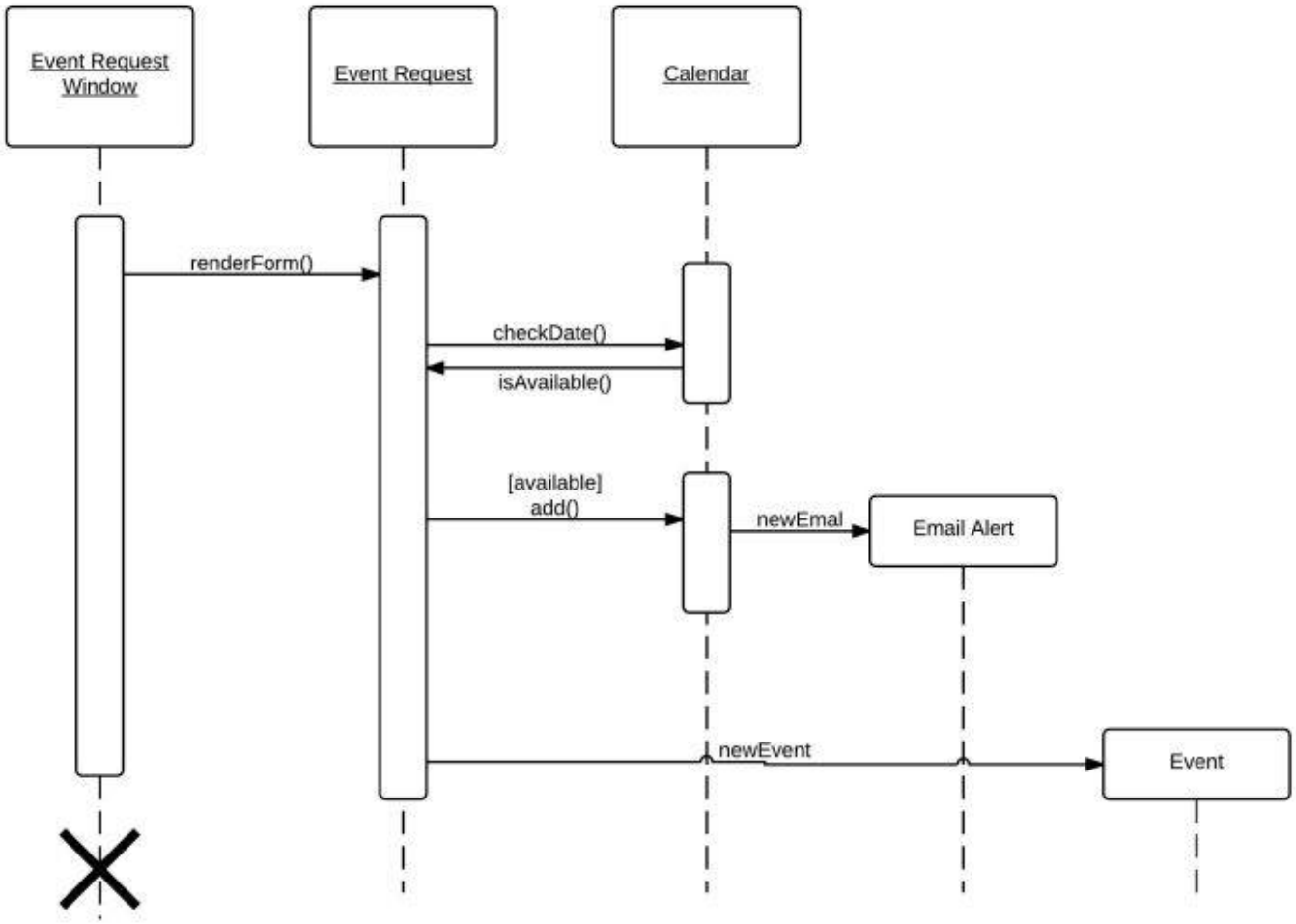
4.6.2 Component

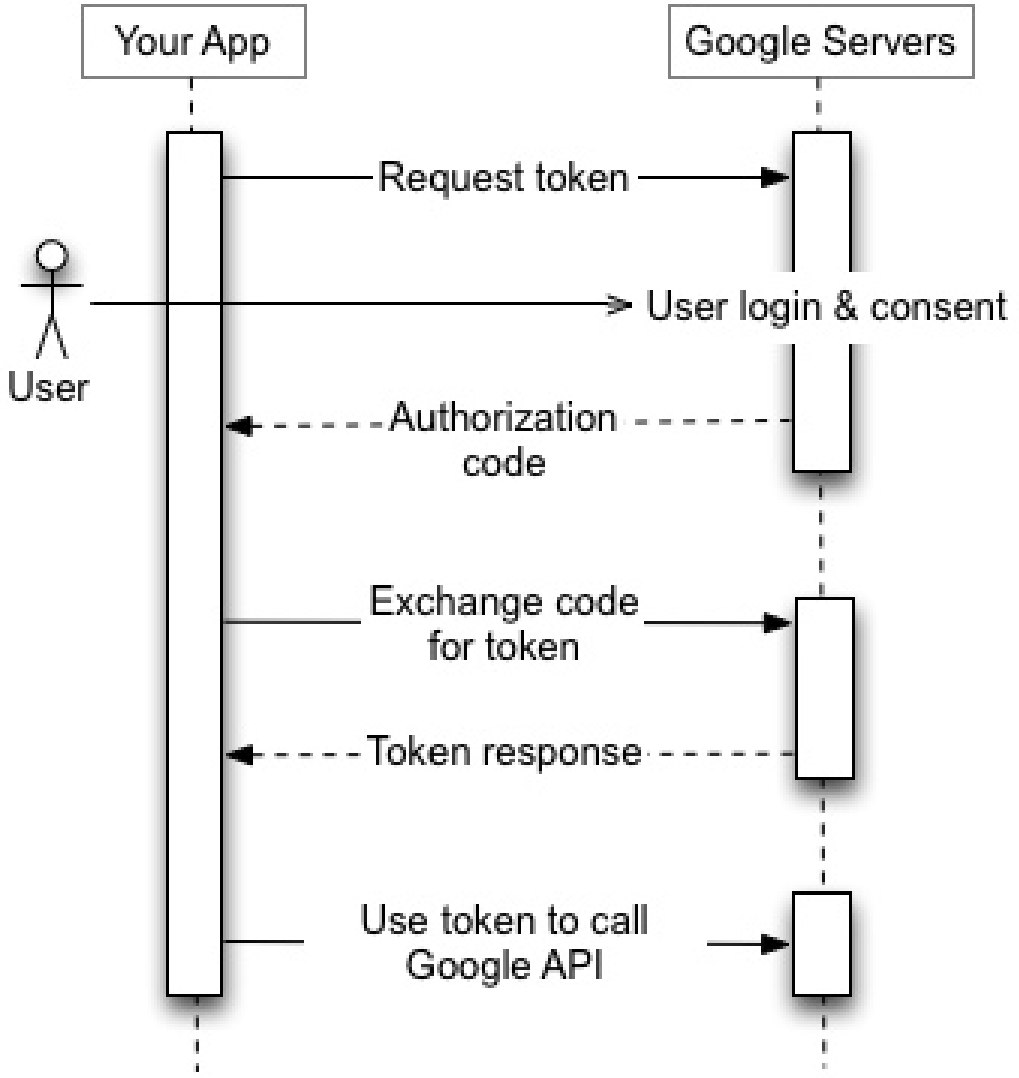


4.7 Sequence


User
Login Screen
Validate User Module
Database
Click Login
Show form
Enter Login/Password
Validate User
Validate User
User Login Succesful
User Login
Current User


User
Main Form
To do Module
Database
Enter To Do Menu
Regular Tools
Find All To Do
Show All To Do's
Return To Do's
Return To Do's
Click Add To Do
Show To Do Form
Submit Form
Save To Do
Save To Do
Show To Do
Success
Success

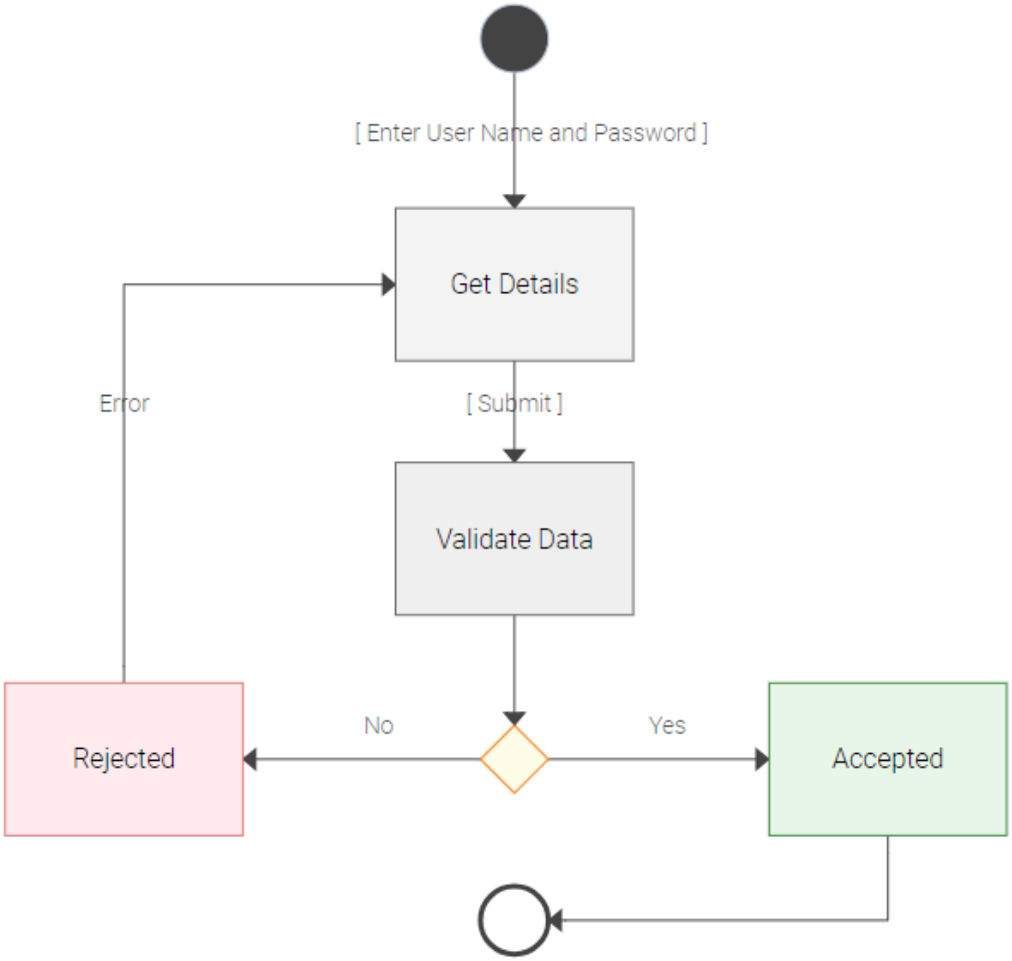

Event Request Window
Event Request
Calendar
renderForm()
checkDate()
isAvailable()
[available]
add()
newEmal
Email Alert
newEvent
Event



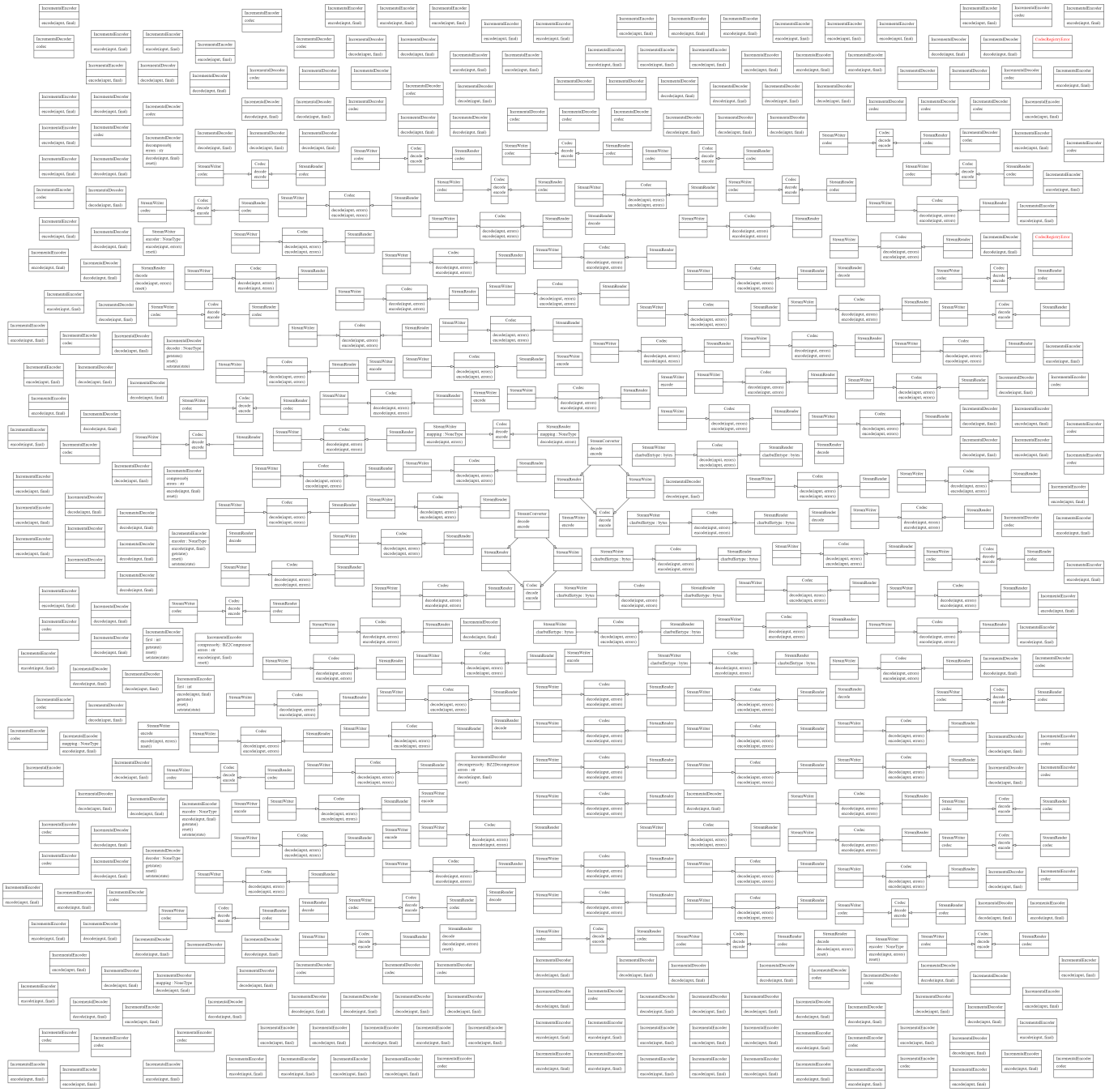
The authorization sequence begins when application redirects a browser to a Google URL; the URL includes query parameters that indicate the type of access being requested. Google handles the user authentication, session selection, and user consent. The result is an authorization code, which the application can exchange for an access token and a refresh token.

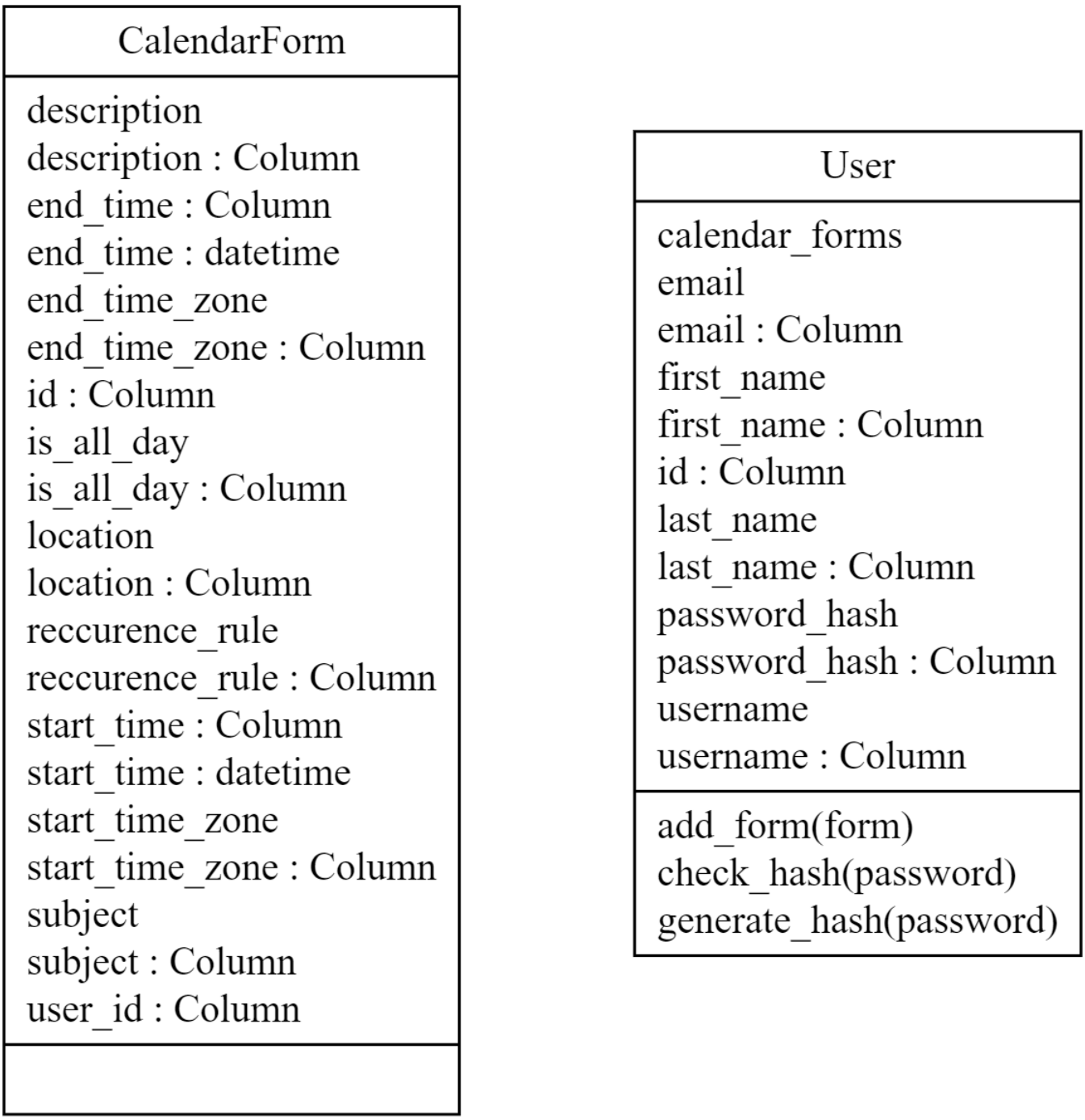
The application should store the refresh token for future use and use the access token to access a Google API. Once the access token expires, the application uses the refresh token to obtain a new one.

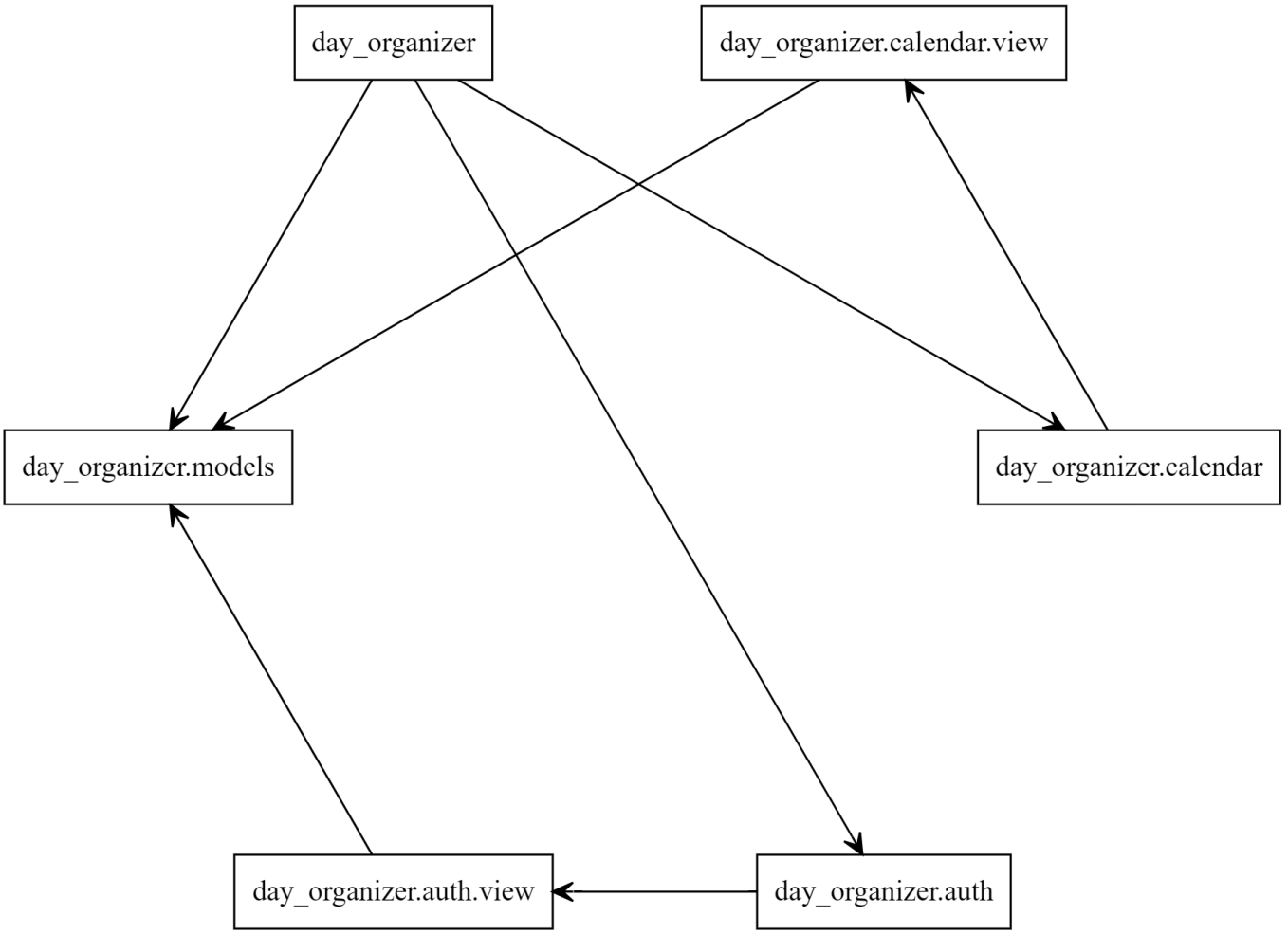
4.8 Activity


[Enter User Name and Password]
Get Details
Error
[Submit]
Validate Data
Rejected
No
Yes
Accepted

4.9 Class




CalendarForm
description
description : Column
end_time : Column
end_time : datetime
end_time_zone
end_time_zone : Column
id : Column
is_all_day
is_all_day : Column
location
location : Column
reccurence_rule
reccurence_rule : Column
start_time : Column
start_time : datetime
start_time_zone
start_time_zone : Column
subject
subject : Column
user_id : Column
User
calendar_forms
email
email : Column
first_name
first_name : Column
id : Column
last_name
last_name : Column
password_hash
password_hash : Column
username
username : Column
add_form(form)
check_hash(password)
generate_hash(password)


day_organizer
day_organizer.calendar.view
day_organizer.models
day_organizer.calendar
day_organizer.auth.view
day_organizer.auth

5 Implementation

5.1 Solution Strategy

5.1.1 Architecture decisions

Our project is based on REST architecture. It consists of client and web server parts. Without much detail, our clients send a request to the server (like GET and POST HTTP requests), and depending on conditions server-side gives a response. For example, the client wants to see some protected page, for example, the client sends a request to the webserver. If the request is valid (for instance, the user is logged in), then we give him access to this part of the info. It is worth mentioning, that the main advantage of REST architecture is in task separation. Generally, all related to UI and Backend is separated into two parts, and interaction is done via JSON or something like XML. Besides, it makes code less chaotic and also more secure.

5.1.2 Technology stack

After a long consideration, we decided to use the next technology stack:

a) html/CSS + React (frontend);

b) Python with flask framework (backend);

c) Nginx + Gunicorn + dokku + drone.io (Nginx - proxy for frontend and gunicorn backend, gunicorn - python wsgi server, dokku - deployment, drone.io - continuous integration)

d) Gitea (similar to GitHub) this is like architecture decision + technology stack

5.2 Project initialization

5.2.1 Task management/distribution

As management tool we choose to use notion.so - mainly because it offers students a pro version, as well as allow us to use the Notion platform as a common blackboard, to-do list and to keep easy track of what everyone has done or what is supposed to do next. Below is a snapshot of our notion dashboard:

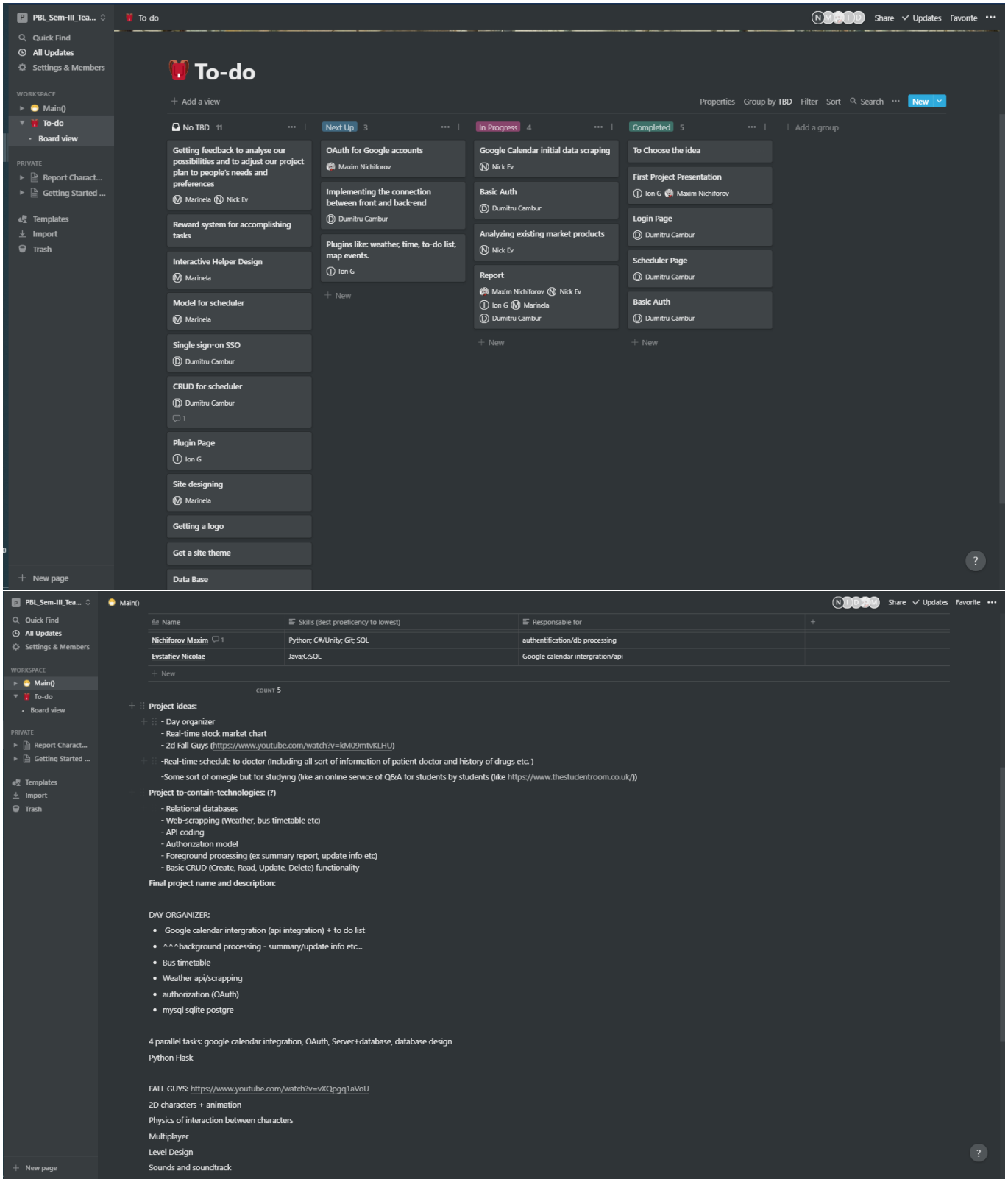


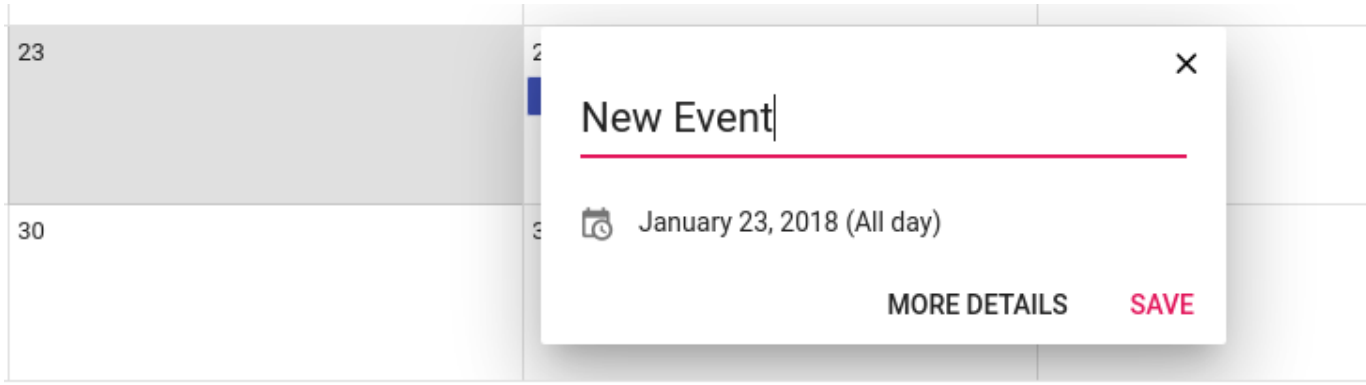
Figure 5.1 – Notion platform used to share tasks

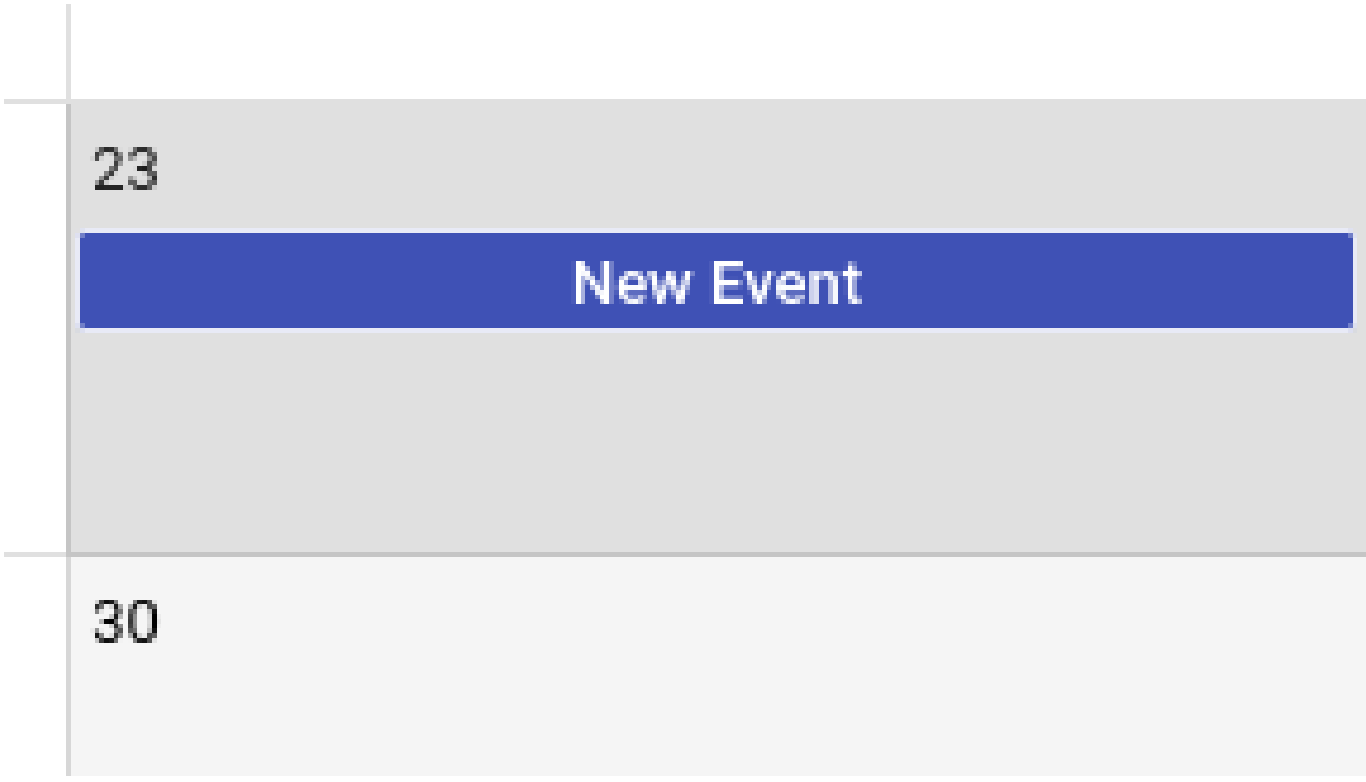
5.2.2 Distributed Version Control of the involved repositories

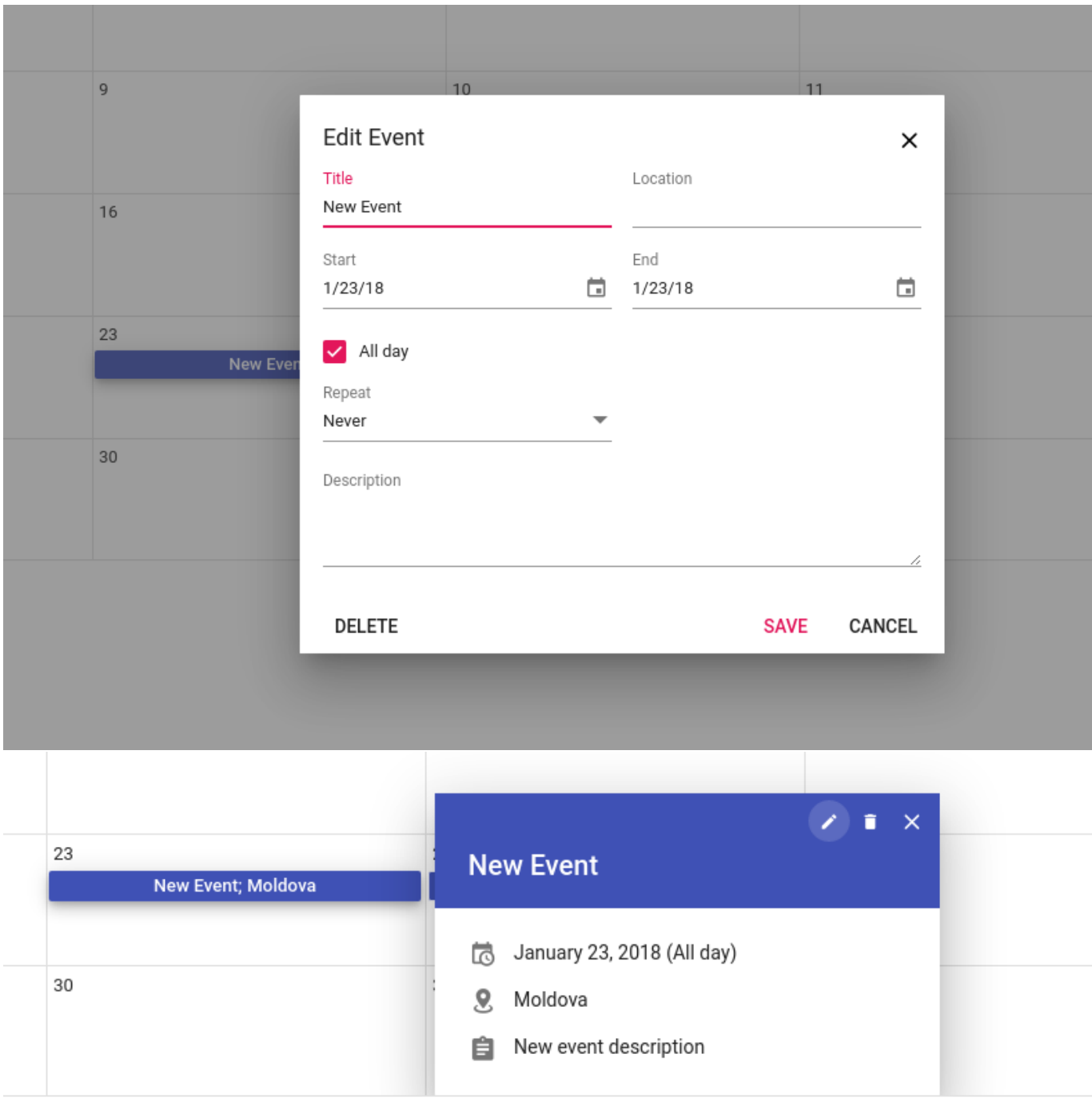
We choose to use Git and to host our code base on Gitea over GitHub, as it offers better overall features that we may need in future upon developing further the project. Gitea is an open-source forge software package for hosting software development version control using Git as well as other collaborative features like bug tracking, wikis and code review. It supports self-hosting but also

provides a free public first-party instance hosted on DiDi's cloud. Source code hosted on Gitea:
Front-end part:`https://git.bitcore.nz/dima/day_organizer_fe`
Back-end part: `https://git.bitcore.nz/dima/DayOrganizer-be`
Initial work done on GitHub (no longer maintained):`https://github.com/waffle4everyone/PBL_Organizer`

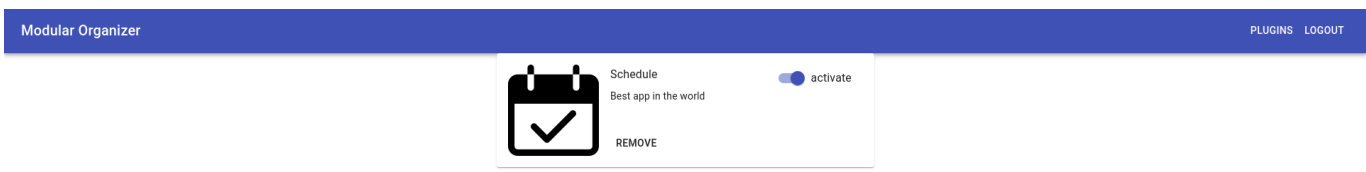
5.3 Screenshots

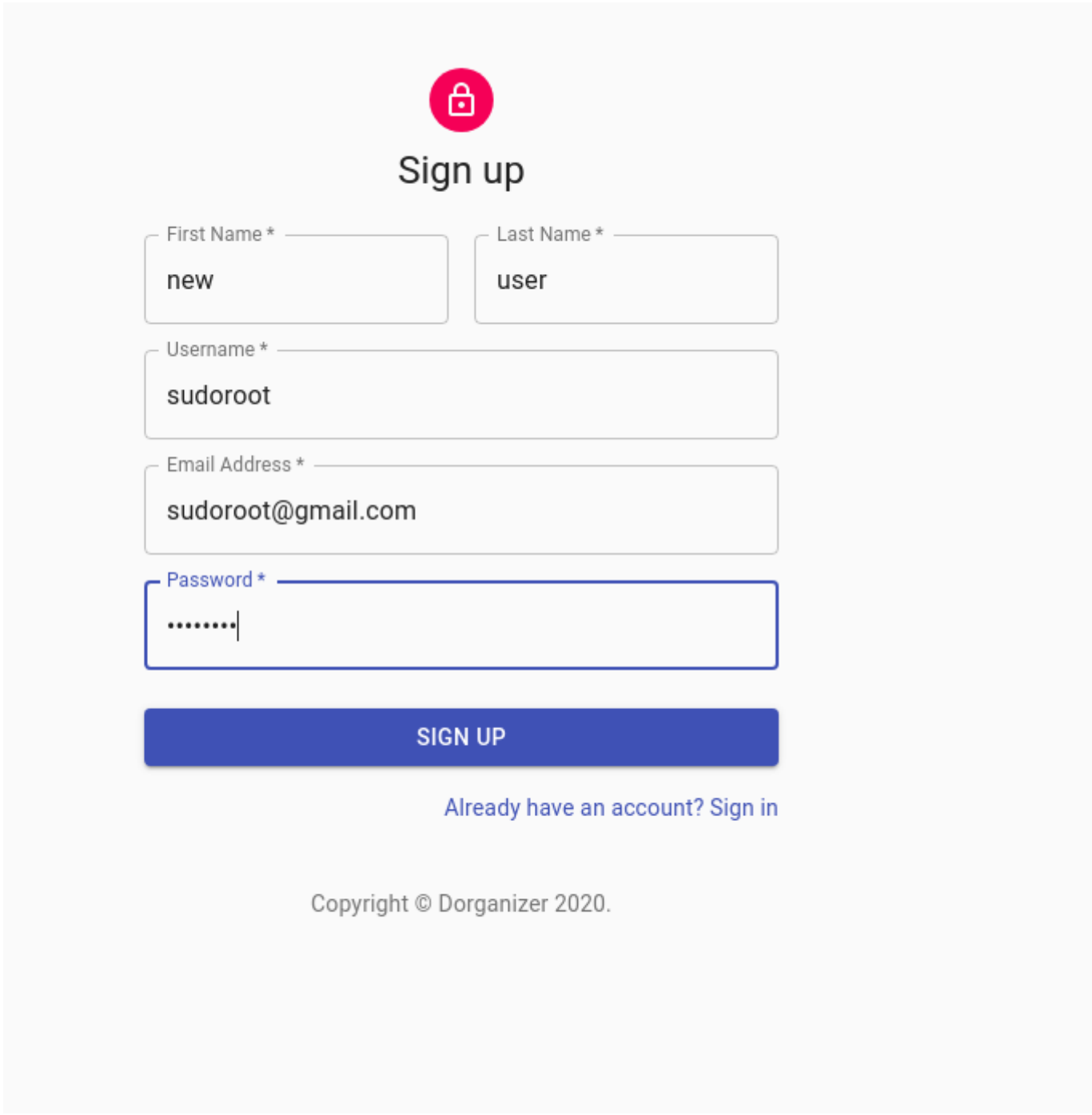


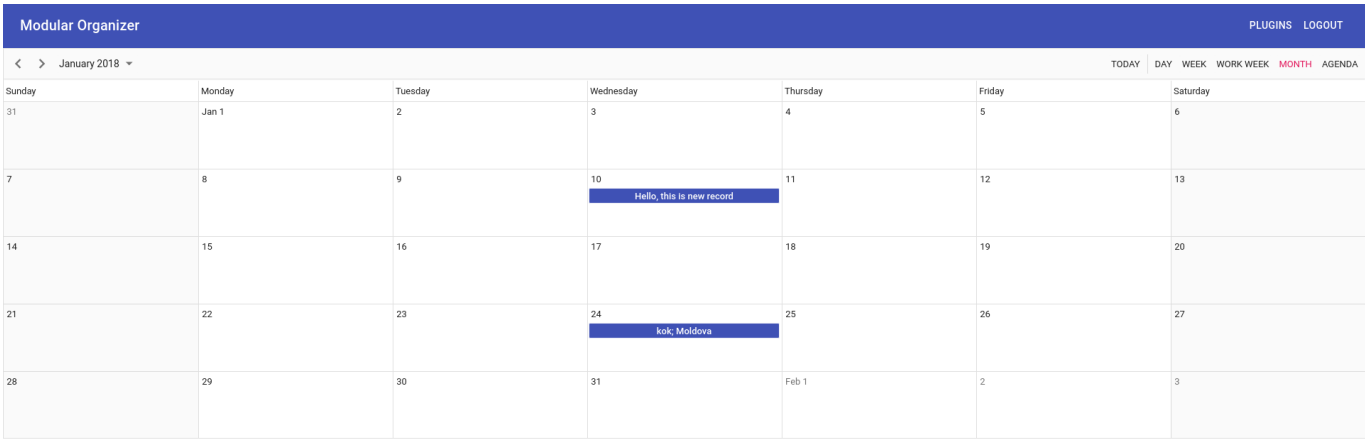


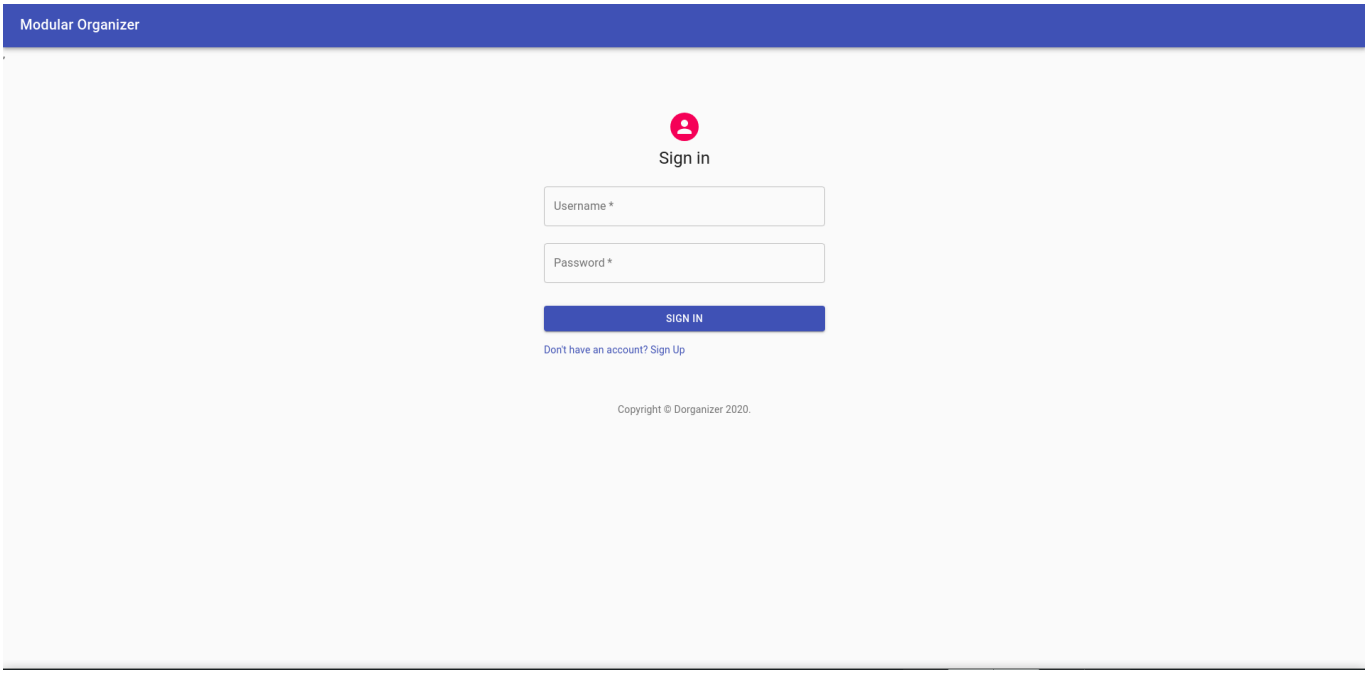

9
10
11
16
23
New Event
30
Edit Event
Title
New Event
Location
Start
1/23/18
End
1/23/18
All day
Repeat
Never
Description
DELETE
SAVE
CANCEL

23
New Event; Moldova
30
New Event
January 23, 2018 (All day)
Moldova
New event description


Modular Organizer
PLUGINS
LOGOUT
Schedule
Best app in the world
activate
REMOVE


Sign up
First Name *
new
Last Name *
user
Username *
sudoroot
Email Address *
sudoroot@gmail.com
Password *
SIGN UP
Already have an account? Sign in
Copyright © Dorganizer 2020.


Modular Organizer
PLUGINS
LOGOUT
January 2018
TODAY
DAY
WEEK
WORK WEEK
MONTH
AGENDA
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
31
Jan 1
2
3
4
5
6
7
8
9
10
Hello, this is new record
11
12
13
14
15
16
17
18
19
20
21
22
23
24
kok; Moldova
25
26
27
28
29
30
31
Feb 1
2
3


Modular Organizer
Sign in
Username *
Password *
SIGN IN
Don't have an account? Sign Up
Copyright © Dorganizer 2020.