Bilkent University Department of Computer Engineering

# Senior Design Project

*Project Name:* Cronus

## Final Report

*Group Members: Gizem Karal, Gökberk Boz, Hüseyin Fatih Karahan, Irmak Çeliker*

*Supervisor: İbrahim Körpeoğlu*

*Jury Members: Shervin Arashloo and Hamdi Dibeklioglu*

*Final Report May 6, 2022*

*This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Senior Design Project course CS492.*

# 1. Introduction

Project management is a very significant step during the creation of any kind of product. Especially in team projects, it can be very challenging to keep track of the project, distribute and order the tasks due to the number of people working in the project. Along with the fact that it can be hard to carry out a project with many different developers, due to the COVID-19[1] pandemic it has become even more complicated to keep track of others' work since project management is also needed to be carried out on digital platforms. Our aim is to solve these kinds of complications and give relief to project developers during their project creation process by creating a project management tool, Cronus, for software developers.

Moreover, besides the complications that arise due to the number of members in a project, it can also be challenging for people to keep track of their own process and work. Cronus will eliminate these problems and create a neat work environment for their users.

Every project in addition to normal tasks also has some repetitive tasks that need to be tracked. While Kanban[2] tools like Jira[3] or Asana[4] allow the tracking and ordering of issues, they become cumbersome very quickly when dealing with tasks that need to be repeated over intervals. Usually this is dealt with by creating a new issue during each sprint or whenever it’s needed. Our project management tool instead focuses on the management of tasks that repeat.

# 2. Requirements Details

## 2.1 Functional Requirements

### 2.2.1. Client-side Web App

- The application should allow users to sign up and login.
- The application should allow users to subscribe and/or change their subscription for the system.
- The application should allow users to create a team to manage the users that will be using the system for their entity, such as a company or a school.
- The application should allow team creator(s) to create roles with different permissions and assign them to the members of their team.
- The application should allow users to create a project.
- The application should allow users to create a board under a project to manage different parts of the project.
- The application should allow users to create tasks that need to be completed in specified intervals.
- The application should allow users to add information to a task, such as a title, a summary, priority etc., or a custom field defined for the project.
- The application should allow users to create different labels for tasks on a per board, per project or per team basis and assign them to actions.
- The application should allow users to assign one or multiple users to a task and define if any given interval will be considered completed if it's done by one of the assignees or if it needs to be done by every assignee.
- The application should allow users to watch a given task to be notified whenever there's a change done to the task.
- The application should allow users to define a prerequisite to a task that needs to be completed before that task can be completed.
- The application should allow users to search, filter and sort through all tasks, boards, and projects they have access to.
- The application should allow users to view the log of all changes done by the users.

### 2.2.2. Backend Server

- The system should store all tasks, boards, projects, teams and all associated information.
- The system should log all the changes done by the users.
- The system should notify users when there's a change in the task they're watching.
- The system should notify users when they're assigned to a task by someone other than themselves.
- The system should process payments for subscriptions and send invoices following the team's subscription schedule.

### 2.1.2. Admin Webapp

- The admin application should allow admins to view and edit all past subscriptions.
- The admin application should allow admins to view all the statistics about the financial data about the application.

## 2.2 Non-functional Requirements

### 2.2.1 Usability

The most important non-functional requirement of the application is usability, since it focuses on keeping track of users' process and work. The application targets to create a neat work environment for its users. To achieve user-friendliness in our design the following points are considered:

The main purpose of the application is to present a neat environment for its users in a team, in order to track and order the issues in the project. The application should present the workload and project plan in a clear display with its user-friendly UI. To ensure this point is satisfied, we plan to use a well defined user interface system such as Material or Ant.

It should be straightforward for users to add information/comments to achieve user-friendliness. To manage that, users will be able to fill in feedback at the end of the project.

### 2.2.2 Reliability

The system should be composed of highly reliable functions with a similar efficiency after extensive use. To achieve that, during the testing process, the application’s critical failures will be checked and calculated. It gives the information of the system according to the failures. If the number of failures is low, that means the system is reliable.

Since the system provides continuous information to users during the project, the system should be robust, which means it should handle crashes in the software with a high level of control and efficiency.

### 2.2.2 Localization

The application has a localization non-functionality which is pretty important from the user’s perspective. A project may consist of members who are from different countries. It leads to different time zones for different geographical locations.

In order to handle it, the system will have aspects such as time zone.The time zone has to be calculated according to each member’s location and schedule will be shown in this sense.

### 2.2.4 Availability

During the project, everyone has their own schedule which leads to different working hours. The users should be able to access the application at any time they want, some may choose the morning scale while the other ones choose to work in the evening.

### 2.2.5 Security

To protect the sensitive data of the user, the application provides password generation and and security question answering:

The  application does not grant access until the user creates a strong password. For example, a strong password might contain a certain number of characters and a capital letter [6].

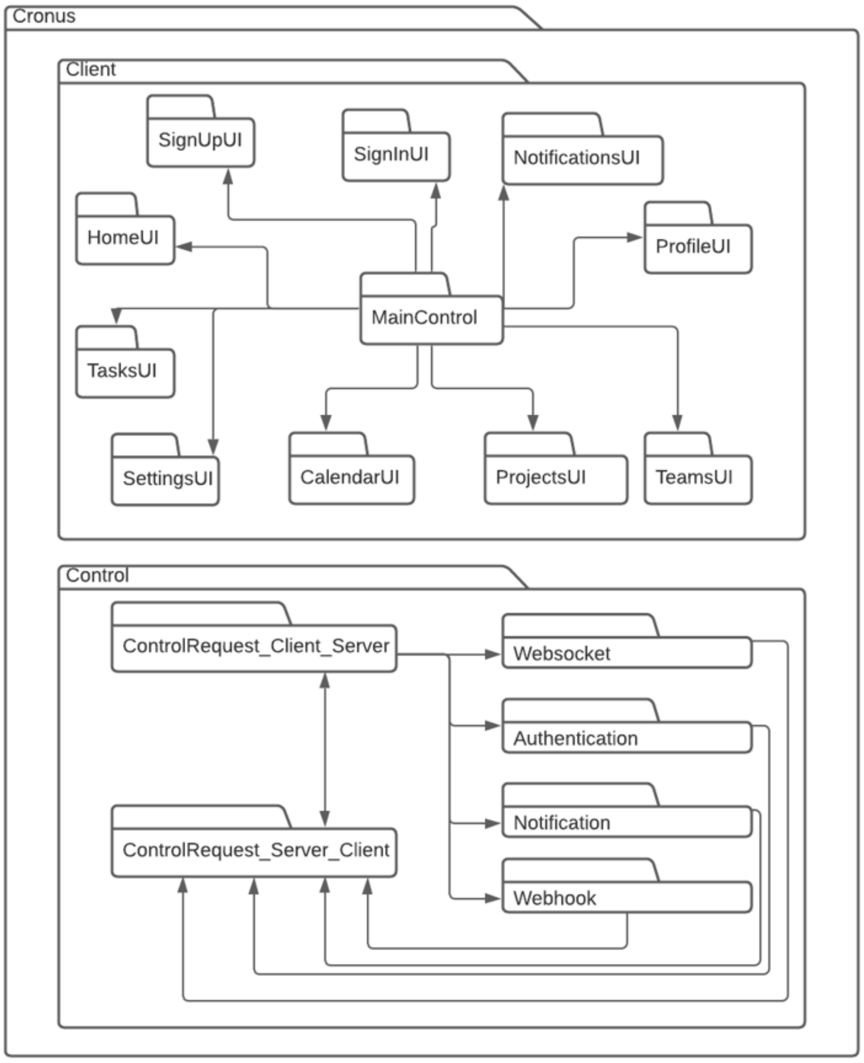
### 2.2.6 Performance

The application will respond to its users in a quick way showing which tasks are completed and will be repeated after finishing it. In order to manage that, the users should be notified with real-time notifications in a fast way within 1 second at most.

# 3. Final Architecture and Design Details

## 3.1 Client Side

The client side of the application is divided into two parts one for the UI interactions and the other for the control of the data input/output done by the user.



### 3.1.1 Client UI

Client UI is the part where the user interacts with the application. The information shown to the user comes from the server to the control side and then to the UI. The data changed by the user also uses the same way to return to the server.

The purpose of splitting Client side into two parts is the UI and rest of the application works in different ways. Putting a control algorithm to UI generally ends up data losses and crashes.

The UI parts of the application can be seen above, we created a component for each important action and entity. The authentication is connected with sign-in and sign-up however data flow must be done for that user in every UI page. The tasks, calendar, projects, teams, profile pages directly use data from the server.

### 3.1.2 Client Control

Client control ensures the connection between client UI and server. Creates necessary responses, requests, notifications or data packages.

- ControlRequest classes create requests and responses between server and ClientUI.
- WebSocket and WebHook sends the updates from the server to the client according to scope.
- Authentication is for the stable connection.
- Notification sends the notifications through in-app or email.

## 3.2 Server Side

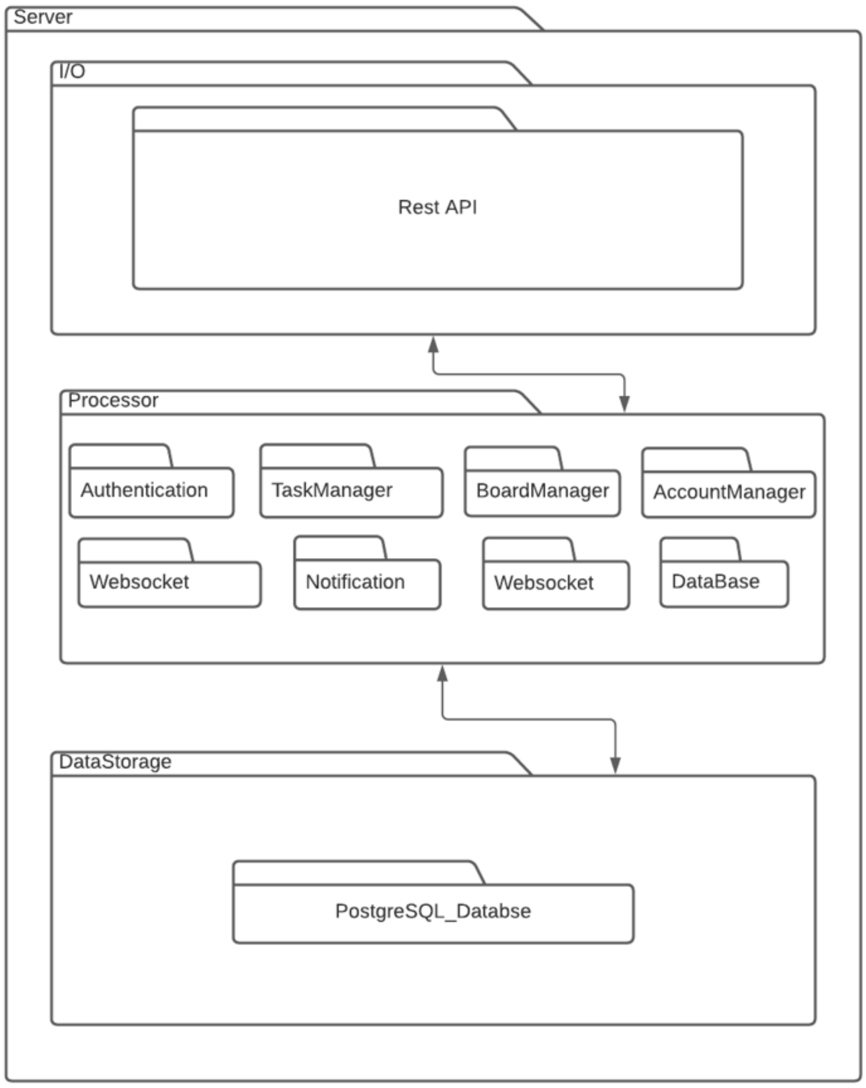
The server side of the application is composed into three parts.

- I/O part uses RestAPI and carries the requests done from client side to the Processor part.

- Processor consists of several packages. These packages maintain and updates the features named.

- Authentication: Authentication for the clients during server access
- TaskManager: Maintains task feature to the client
- AccountManager: Maintains account management for the client
- BoardManager: Maintains board feature to the clients
- Webhook: Publish updates to the client.
- Websocket: Send live updates to the client.
- Notification: Send notifications to the client.
- DataBase: Maintains management of the database queries

- DataStorage is the database. Used PostgreSQL.



# 4. Development/Implementation Details

## 4.1 Authentication

Authentication in Cronus is handled by using a centralized authentication authority, and by storing remaining personal information in our application. In our case, we chose Firebase to handle our authentication needs. When users first register, they enter their details in the client and then the client creates an account in our Firebase system. For an account to be recognized however, it also needs to exist in our backend database. To handle this, after creating a user the client sends a request to the UserController, which registers the UID sent by firebase together with the information. From that moment on, whenever a request comes to the server, the UID in the token is checked to see if it exists in the database, and if it doesn't the application returns a 406 error, asking the user to register through the UserController.

For sign in, users sign in through the login page on the client, which uses the official firebase package to sign in. Afterwards, the idToken returned by firebase is added to every request sent to the server in the Authorization header as a Bearer token.

## 4.2 Repeated Tasks

Repeated tasks are handled through the use of a master-copy relationship. First, a master entity defining the information is recorded to the database. The master copy doesn't contain any concrete dates, only the length this task will need to be completed in, like 3 days or 1 week. After creating the master entity, a copy is created that points to the master entity. This copy is called an EphemeraTask. Each ephemeral object contains a list of members that needs to complete the issue to keep track of which members completed this task in each iteration, and start/end dates that defines the range this iteration will occur.

Whenever a task is marked done by someone, the AssignedMember in that iteration is updated as done by the TaskService. Afterwards, an email is sent to every watcher in the master copy containing the completion details.

A worker checks the currently existing jobs at regular intervals, and creates a fresh copy whenever a task reaches its end date. For the worker intervals, we have chosen an hour so that the checks that run through the whole issue database don't coincide with the next run of the worker.

## 4.3 UI

The UI is a React application. It is built with reusable components, and pages where each component is used as needed. Additionally, there are utility functions that are used for common interactions in the application such as authentication, getting the user details, sending requests etc. The react application uses the newer, more modern hooks based system instead of the class based system to make sure the application will have official support for longer, and to avoid any bugs that are possibly living in the React codebase. The UI uses client-side routing via React Router. This package allows us to modify the route as needed when switching components to provide better user experience. It also allows us to limit access to some routes, depending on the permissions of the user. Requests to the server are handled by the axios package. After a user logs in, this package allows us to create an interceptor that runs before a request is sent and add the necessary headers to the request.

Theming of this application is handled via Syncfusion and MaterialUI. Both offer battle-tested components that can be used with confidence, as well as a high degree of customization. The reason behind choosing Material UI over other design systems is that Material UI is used by a large number of applications around the world. It is also the official design system of Google, thus applications built with Material UI looks and behaves similar to applications millions of people use all the time, increasing the user experience and decreasing the time spent to get familiar with Cronus.

## 4.4 Webhooks

In addition to the official client, our application also has a webhook endpoint that users can subscribe to see the changes and connect their own applications to our application if needed. Access to webhooks needs authorization that can be obtained by the organization owner by generating an api_access_token. With this token, anyone can connect to the webhooks and listen to the events.

# 5. Testing Details

Cronus is not a big-scaled project however, it is more complicated than many of our previous works. This complexity makes the testing process more important. Since there are more connections between classes, data flow is vital, UI pages that may overlap etc. We used different testing patterns in order to ensure the application is going to run flawlessly.

## 5.1 Continuous Integration

The development process of Cronus was not advanced so strictly like in the waterfall method. Each member of the group has different responsibilities and fields to work on. In order to maintain this process and see/check or benefit from each other's work we used many features of GitHub. Push/pull, merge, creating different branches properties of GitHub helped us to create an error-free mainline. This line contained the working parts of our application from different members. Also this approach made our work easier while combining the different pieces of our application. Also the ensured line is pulled and tested by everyone, detecting an error earlier increased the quality of our code. Additionally, each push to the master branch automatically deployed the application to Netlify, allowing us to test the application in a production environment.

## 5.2 Code Review

Code review was an essential part while working separately. One must write the code with proper names, scopes, use the same APIs and libraries. With this way other members could check and use the new code pushed by other members. Determining those features before starting writing may speed up the whole development process, since members will check and use each other's code and may offer better ways of writing it.

## 5.3 UI Testing

Cronus must be user friendly and easy to control and understand, like other applications that interact with users much. The user side of the application must be clear and well organized, every important feature must be reached easily without problem. Also the authentication and redirection in pages are essential since we have a strict hierarchy inside the application. So every UI page and their connection must be tested. Testing involves the accessible pages from different users and also whether the correct information is shown to users or not.

## 5.4 Server Performance and Data Flow

The importance of data flow and performance is stated above. Cronus heavily depends on the constant data flow and change. We try to test each data package multiple times with different test scenarios to detect any bugs or defects. In order to serve a flawless application we will continue to test those scenarios from time to time.

# 6. Maintenance Plan and Details

Maintenance and improvements will depend on users using the application and providing feedback to us based on their experience. After receiving feedback and reports from testers and users, the application will be updated for a better use. Maintenance will focus on increasing the performance and fixing bugs.

## 6.1 Maintenance

Cronus is a platform that requires constant data flow and change between servers and users. This flow must be constant and stable. In order to maintain this situation, the application must be checked, fixed and updated on a regular basis. This maintenance is vital for Cronus since the many features of the application heavily depend on data flow and share.

## 6.2 Problem Handling

Our application will not be sold in a platform or web-market at the beginning but it can be tested by different companies, teams or individuals for a limited time. Through this testing period the feedback of the users is essential for us. Since selling or hiring the application is full of problems and performance issues will definitely have a bad influence when we start selling the application or renting it monthly.

## 6.3 Other Platforms

Cronus will be a web-oriented application in release. However, today many management apps use mobile platforms parallel to web-applications to increase the application’s usability and audience. This is also our plan for Cronus so that we may increase the field of the application and find new customers.

## 6.4 New Versions/Updates

As the developers of Cronus, we planned many new features for our application. However, first we must see that our promises work well in a company or team environment. Our priority is fulfilling the basic features without any bugs and problems.

# 7. Other Project Elements

## 7.1.Consideration of Various Factors in Engineering Design

### 7.1.1 Public Health

The application can be used in every environment with the internet, so people from different places can create and manage a project together using this tool. Due to the COVID-19 pandemic[], Cronus aims for public health by minimizing the need for project team members to meet face to face.

### 7.1.2 Economic Factors

Cronus aims to increase effectiveness and reduce wasted time, so the application can help individuals or companies to profit indirectly and the employment rate may increase. Application will not require a monthly or one-time payment. Every user will be able to access the same features and everything will be free.

### 7.1.3 Environmental Factors

Using cloud based services to deploy our application allows us to decrease our carbon footprint by lowering the digital waste generated while our application is operational.

### 7.1.4 Societal Factors

By easing up the process of project management, Cronus re-arranges many social organizations such as meetings, presentations, reports. With that Cronos reduces the chaos and conflicts that can arise between co-workers, teammates, bosses, employees, etc. As a result, an increase in social peace can be obtained.

### 7.1.5 Cultural Factors

Since Cronus will be able to be used all around the world, it aims to bring different cultures together. People from different countries will have the chance to create and manage a project together, which helps people to get to know other cultures.

### 7.1.6 Public Safety

The information about the companies and employees can only be accessed by the people who work in that company but according to the hierarchy inside the system. Through this, the application prevents undesirable accesses.

| | Effect Level | Effect |
|---|---|---|
| Public Health | 9 | Physical health |
| Economic Factors | 8 | Increase in profit and employment rate |
| Environmental Factors | 0 | None |
| Societal Factors | 7 | Socially peaceful environment |
| Cultural Factors | 5 | Bringing different cultures together |
| Public Safety | 7 | Prevents third-party and unauthorized access |

*Table 1: Factors to consider during the design.*

## 7.2.Ethics and Professional Responsibilities

In the application, other than the team members, any other user will not be able to see the project or its details in order to maintain projects' privacy. Projects will remain confidential between the team members to prevent situations such as stealing of ideas, copying work, etc. Also, to avoid any kinds of copyright issues, open source products will be used during the development process.

## 7.3.Judgements and Impacts to Various Contexts

The first decision was the language of the application. We chose to implement it in English since many engineering companies prefer it. Besides the equipment and documentation in engineering is in English generally, the international employees must be involved in Cronus.

Cronus will have a subscription system, all the features will be available to those subscribed. However, we plan to give free trys in order to take feedback and see how people use it.

As stated many times, Cronus's main purpose is to avoid confusion in the work environment especially for newcomers. The application will help to organize and learn the work.

## 7.4 Teamwork Details

### 7.4.1 Contributing and functioning effectively on the team

In order to make everyone attend the project, we have study modules and every one is responsible for one of them as a leader. Every schedule has to be led and all group members have taken the one which they are familiar with according to the subject. In each module the leader will decide on the process and the resources needed for that module.

### 7.4.2 Helping creating a collaborative and inclusive environment

All team members should care for each other in order to create a collaborative environment. It is essential to act as a team and complement each other in weak areas to overcome difficulties. Giving a chance to every group member about an idea or perspective for the project is important to increase the performance and team positivity.

### 7.4.3 Taking lead role and sharing leadership on the team

In order to give responsibility to every member, the project is divided into pieces and each member is a leader in that part of the project. Also sharing the updates and sources was an important step for us.

### 7.4.4 Meeting objectives

Before the implementation of the application we spent time on analysis and design stages. Constructed a plan for the development, shared the important corners of the application. Through the development process we tried to stick to this plan.

Our priority while implementing the application is to fulfill the identical features of it. The patterns we used, the UI we designed have changed many times. However, we protected the main features of the application. Cronus includes the features for a management app for repetitive tasks. Such as, company hierarchy, task assignment and being able to change those assignments according to requirements (time intervals, exclude/include people, notifications).

There are features we wanted to implement but couldn’t successfully do. The e-mail notification is one of them and the second was implementing it as an JIRA extension. The first one can be achievable within the near future but the second needs much more effort.

**Teamwork**

During the analysis and design stages, we discussed how similar applications work, which APIs are useful, and what kind of design patterns are needed. Each member contributed to each report. Also the important design decisions in the report done together.

In the implementation stage we divided the work according to interests and knowledge. Done daily meetings online and used GitHub efficiently to track, use and correct, if needed, each other's work.

## 7.5 New Knowledge Acquired and Applied

During the development process, we decided to use a system that would help us develop the application as a team. We used docker containers to deploy any dependencies our application needed such as PostgreSQL, and we used a customized .env file to store any common secrets that needed to be shared between our devices. We also learned how to use scheduled workers with the Hangfire library. This library is used for creating new task copies in a timely manner as needed.

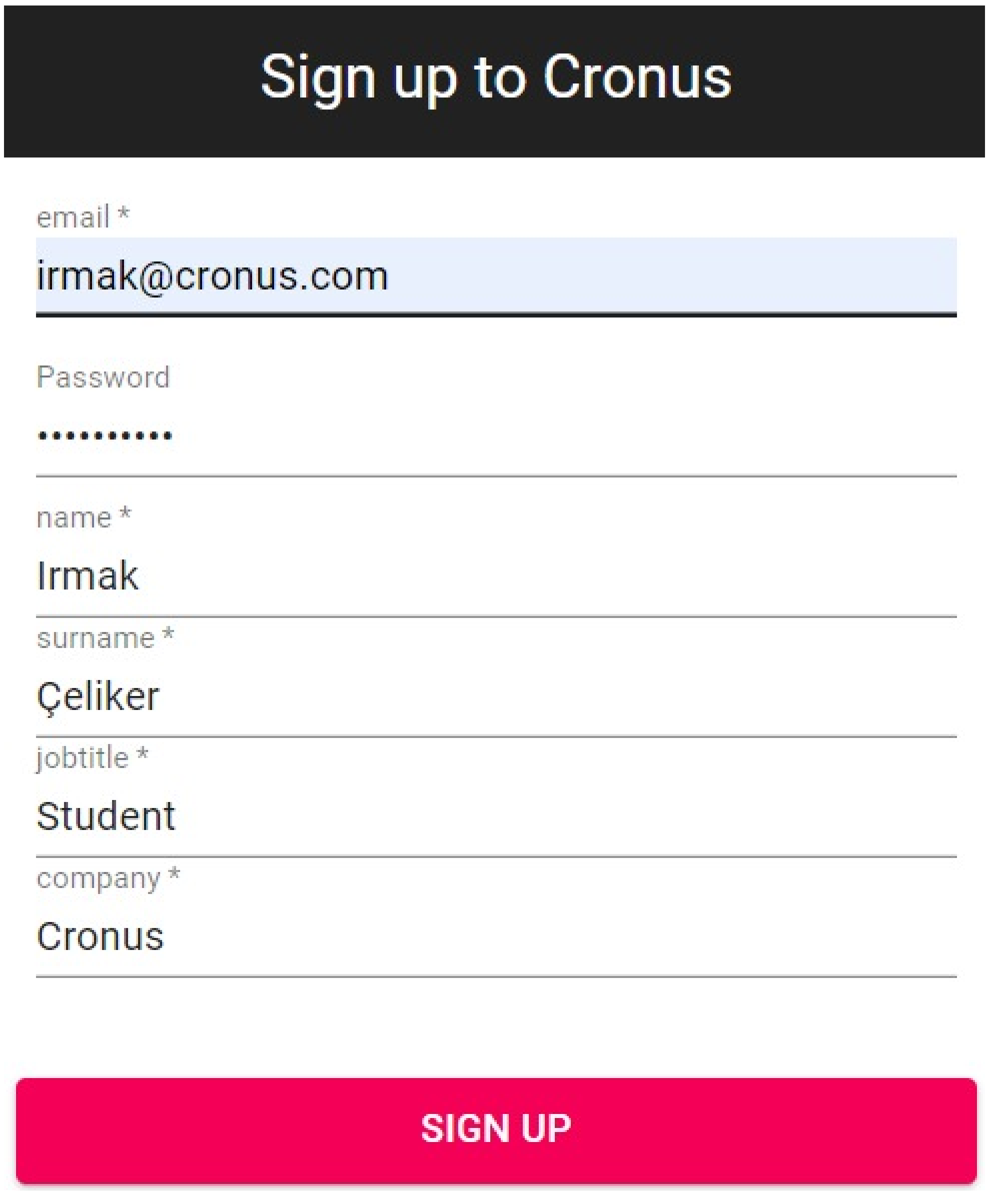
# 8. Conclusion and Future Work

To conclude, Cronus is a work management web-application specializing in repetitive group contents such as; daily meetings, continuous reports, weekly presentations etc. We tried to solve a problem we faced in our work experience.

The first future innovation is bringing the application to mobile platforms and connecting web and mobile platforms to each other for better communication and follow-up. This will increase the range and audience of the application. Since it might be used in daily life or other work fields. Adding extensions with that purpose is the second wish of developers.

# 9. User Manual

**Sign Up**



Sign up to Cronus

email *

irmak@cronus.com

Password

••••••••••

name *

Irmak

surname *

Çeliker

jobtitle *

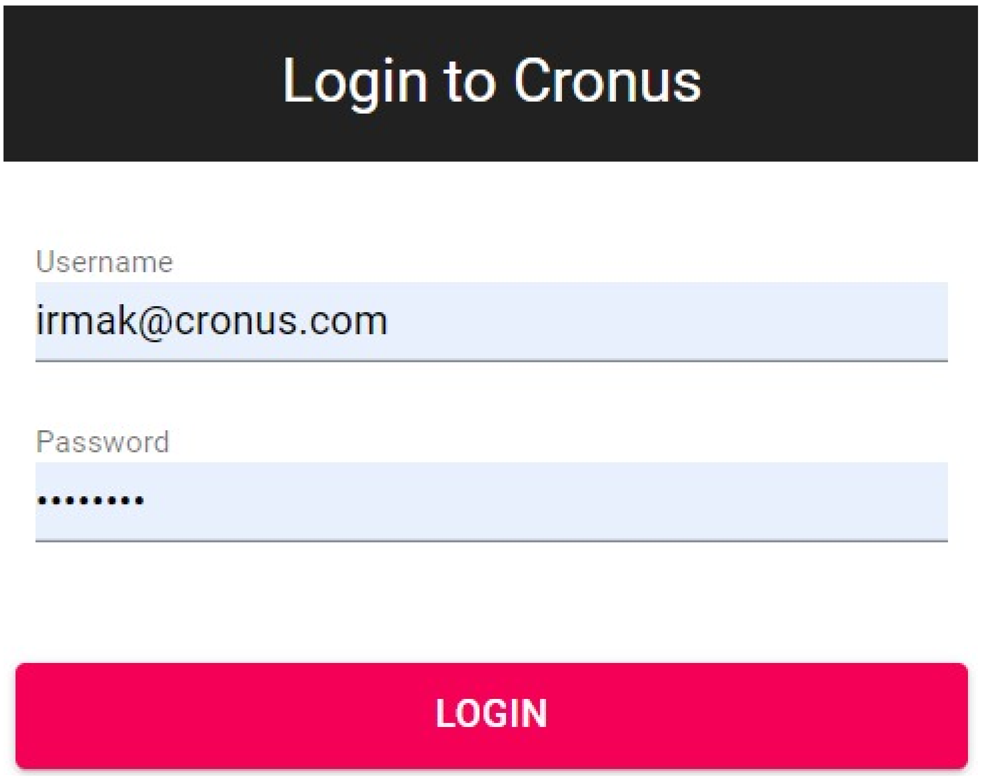
Student

company *

Cronus

- Before signing in, the user has to register and sign up for the application.
- They need to write their email address, password, name, surname, job title and company in the corresponding fields.
- After they are written, they can be loaded on the system after clicking the “sign up” button.

**Log In**


Login to Cronus

Username

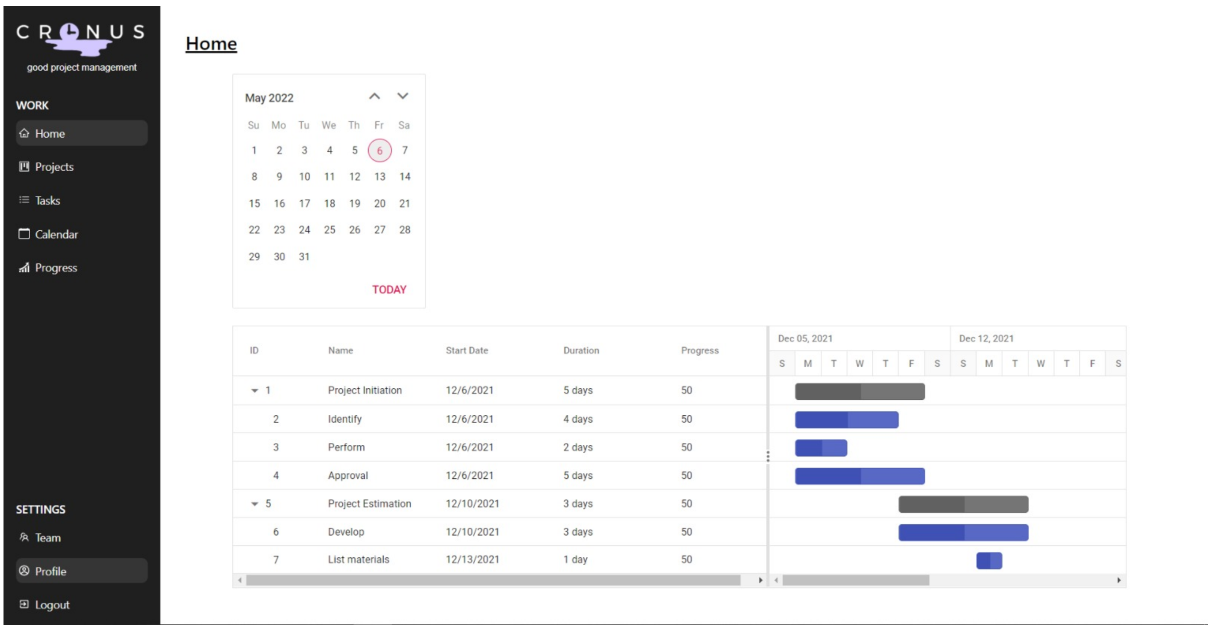
irmak@cronus.com

Password

••••••••

- The application starts with a sign-in page.
- Users can log in to their account by using this page.
- They need to write their email and password in the corresponding fields and press the “Login” button.

## Home

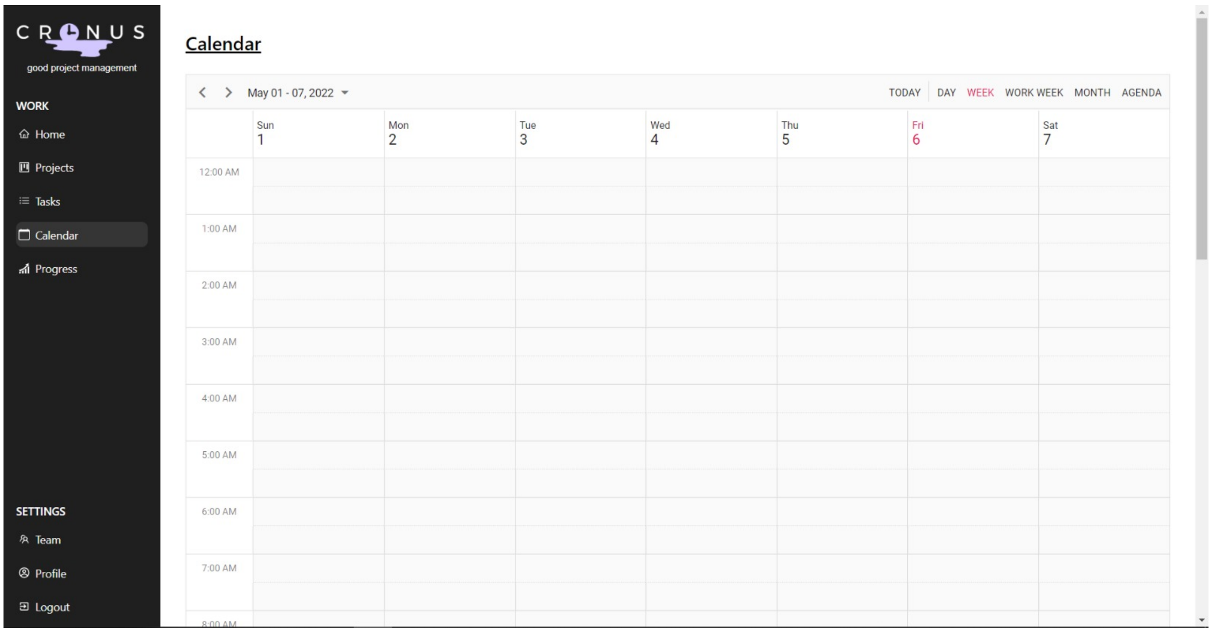


| Su | Mo | Tu | We | Th | Fr | Sa |
|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| 15 | 16 | 17 | 18 | 19 | 20 | 21 |
| 22 | 23 | 24 | 25 | 26 | 27 | 28 |
| 29 | 30 | 31 | | | | |

| ID | Name | Start Date | Duration | Progress |
|---|---|---|---|---|
| 1 | Project Initiation | 12/6/2021 | 5 days | 50 |
| 2 | Identify | 12/6/2021 | 4 days | 50 |
| 3 | Perform | 12/6/2021 | 2 days | 50 |
| 4 | Approval | 12/6/2021 | 5 days | 50 |
| 5 | Project Estimation | 12/10/2021 | 3 days | 50 |
| 6 | Develop | 12/10/2021 | 3 days | 50 |
| 7 | List materials | 12/13/2021 | 1 day | 50 |

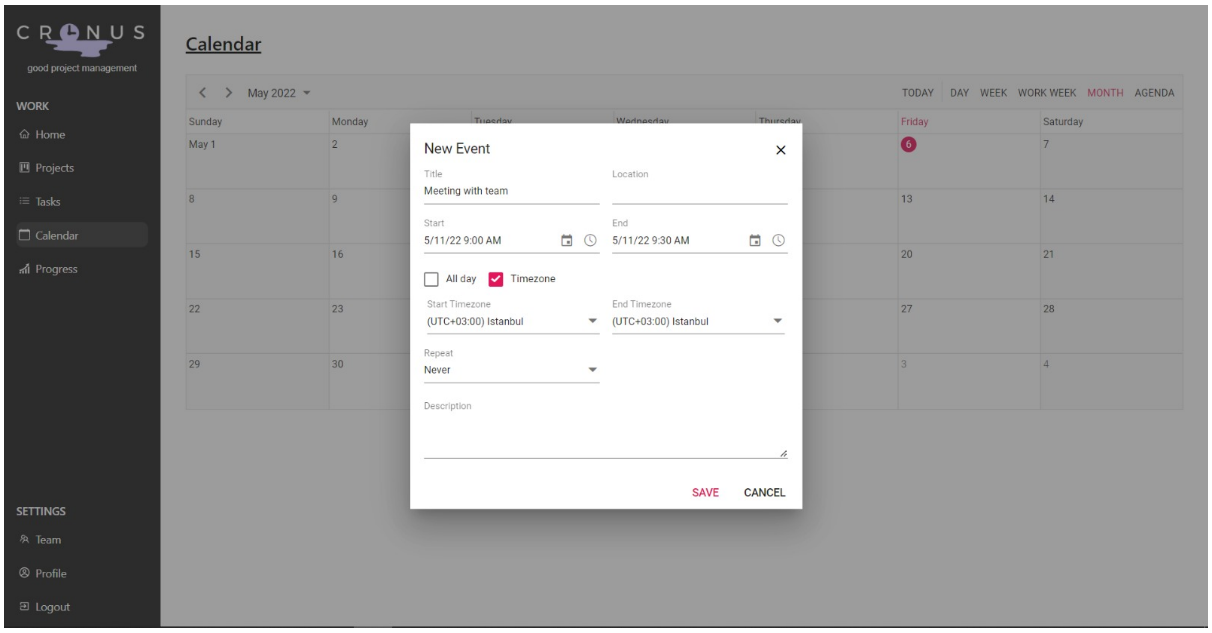
- After logging in to the system, the user can see the day on the agenda section.
- the tasks name, their id names, start date, duration and progress so far can be seen.
- Next to each task, the duration period can be seen easily.

## Calendar



- In the calendar page, the user can see the issues added to her calendar.
- In the options as day, week, month and agenda and work week the user can see the daily or weekly scheduled issues in the project.
- New events can be added after clicking on the box for the date wanted.

## New Event Add Page



New Event

Title: Meeting with team

Location:

Start: 5/11/22 9:00 AM

End: 5/11/22 9:30 AM

All day [ ] Timezone [✓]

Start Timezone: (UTC+03:00) Istanbul

End Timezone: (UTC+03:00) Istanbul

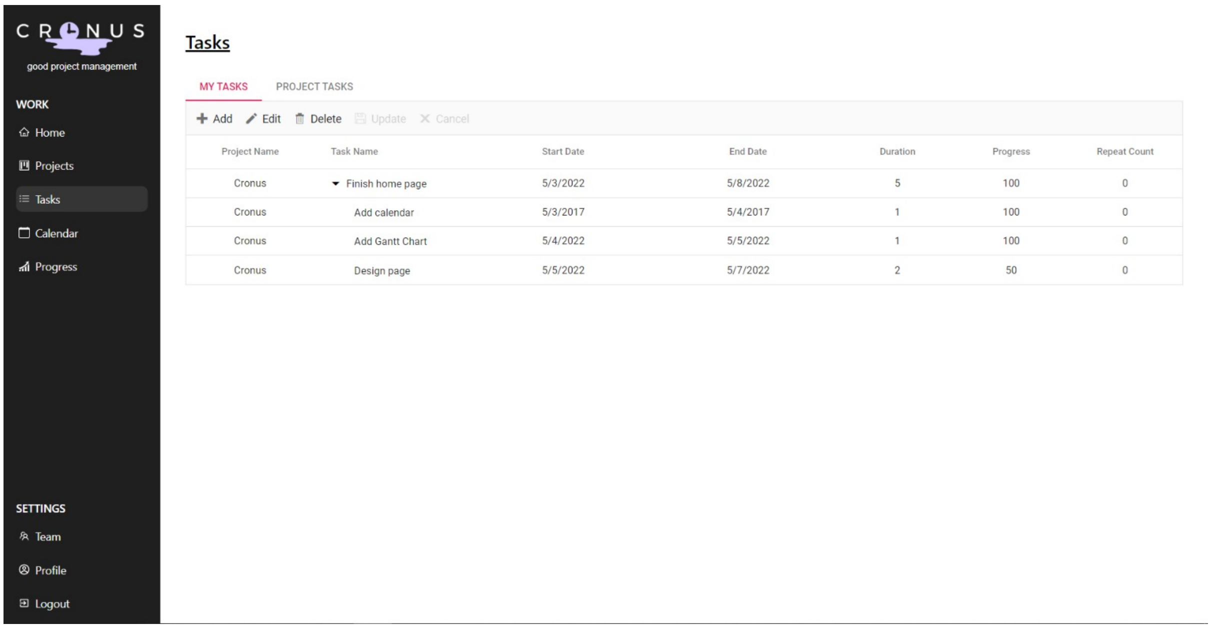
Repeat: Never

Description:

- In the new event page, new events can be added to the calendar .
- Time zones can be set according to the start and end time zones.

- Title can be set from the title part.
- If the task is a repetitive task, in the repeat section it can be determined.
- From the description part the event description is written and it can be saved with the “save” button or it can be canceled.

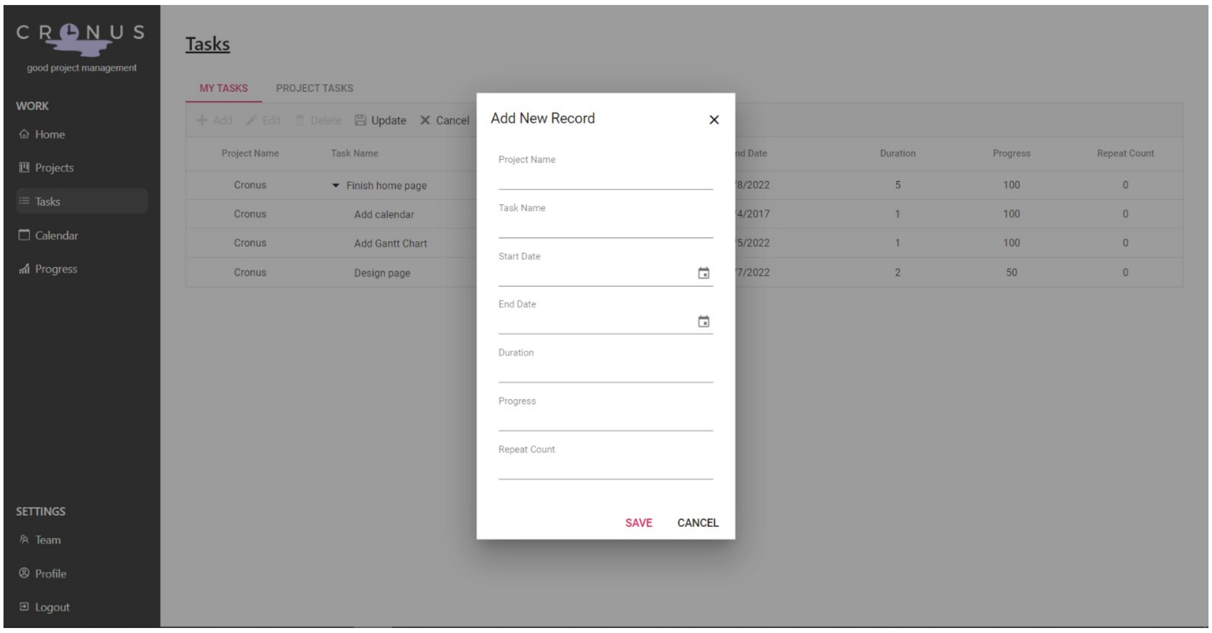
**Tasks**



| Project Name | Task Name | Start Date | End Date | Duration | Progress | Repeat Count |
|---|---|---|---|---|---|---|
| Cronus | Finish home page | 5/3/2022 | 5/8/2022 | 5 | 100 | 0 |
| Cronus | Add calendar | 5/3/2017 | 5/4/2017 | 1 | 100 | 0 |
| Cronus | Add Gantt Chart | 5/4/2022 | 5/5/2022 | 1 | 100 | 0 |
| Cronus | Design page | 5/5/2022 | 5/7/2022 | 2 | 50 | 0 |

- In the tasks page, the user can see each task's progress and details of them.
- If she wants, she can edit, add or delete a task.
- When there is a new task to be added, add new record page is routed.

## Add New Record



Tasks

MY TASKS | PROJECT TASKS

Add | Edit | Delete | Update | Cancel

| Project Name | Task Name | nd Date | Duration | Progress | Repeat Count |
|---|---|---|---|---|---|
| Cronus | Finish home page | /8/2022 | 5 | 100 | 0 |
| Cronus | Add calendar | /4/2017 | 1 | 100 | 0 |
| Cronus | Add Gantt Chart | /5/2022 | 1 | 100 | 0 |
| Cronus | Design page | /7/2022 | 2 | 50 | 0 |

Add New Record

Project Name

Task Name

Start Date

End Date

Duration

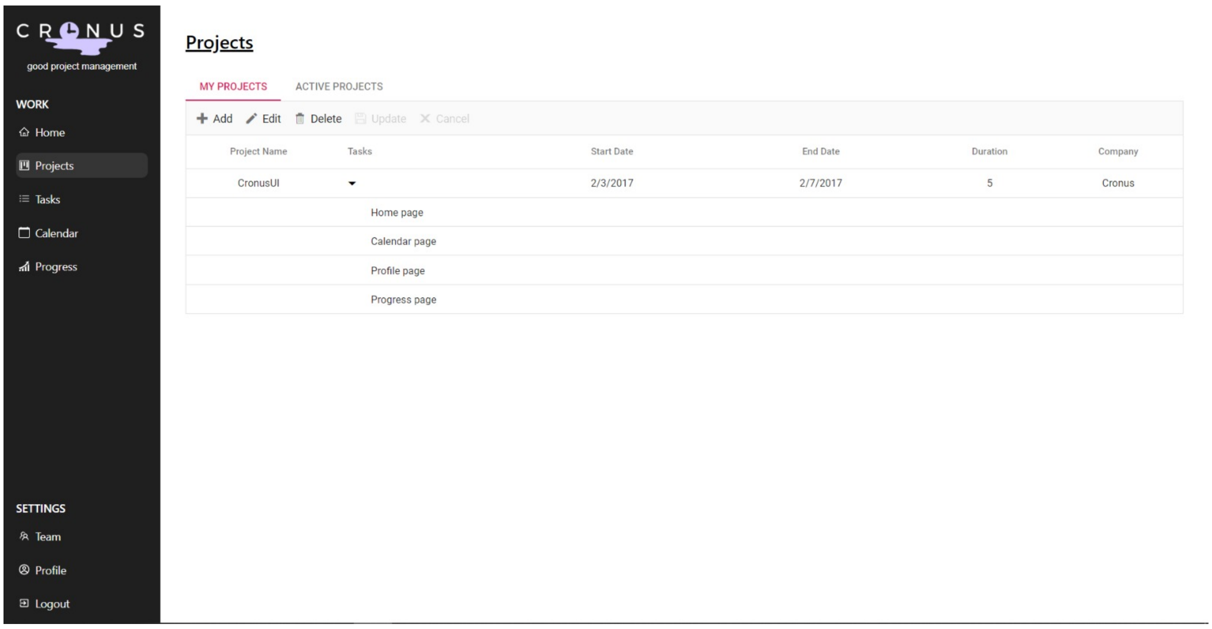
Progress

Repeat Count

SAVE CANCEL

- In the tasks page, add a new record screen is represented.
- Project name, task name, start-end date, duration, progress and repeat count are written and after that they are added to the system with the “save” button.

## Projects



Projects

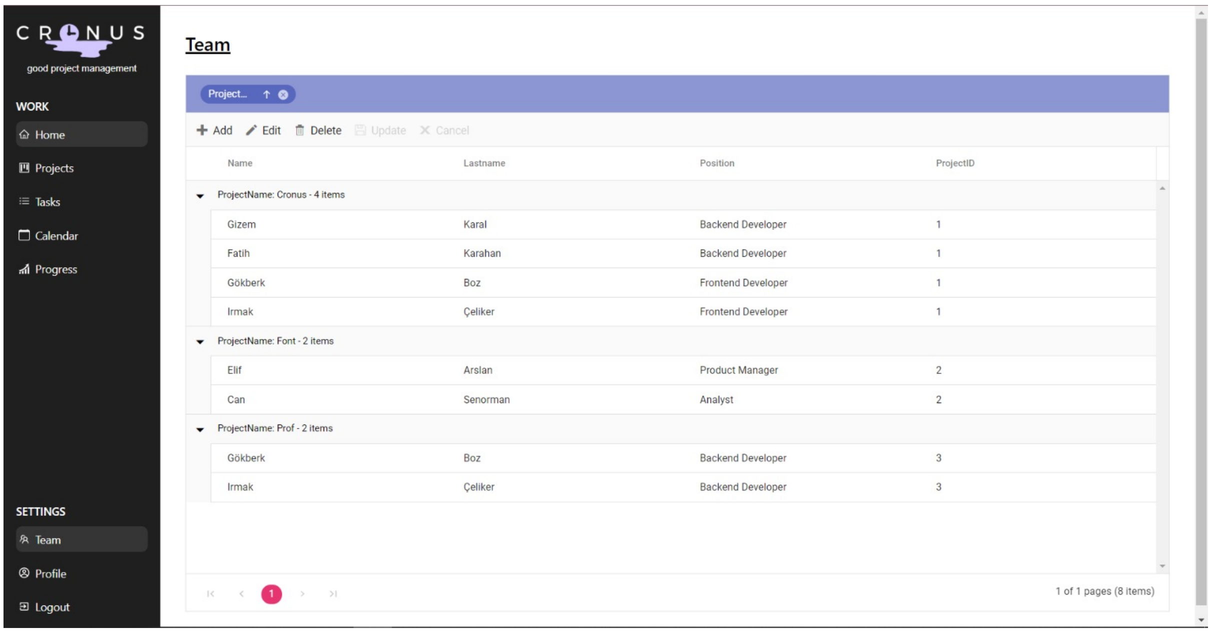
MY PROJECTS | ACTIVE PROJECTS

Add | Edit | Delete | Update | Cancel

| Project Name | Tasks | Start Date | End Date | Duration | Company |
|---|---|---|---|---|---|
| CronusUI | | 2/3/2017 | 2/7/2017 | 5 | Cronus |
| | Home page | | | | |
| | Calendar page | | | | |
| | Profile page | | | | |
| | Progress page | | | | |

- In the projects page, the user can see the projects and active projects of her.
- Each project can be added, edited or deleted .
- When there is a new project to be added after the “add” button on progress, the new add record page is shown.

## Team

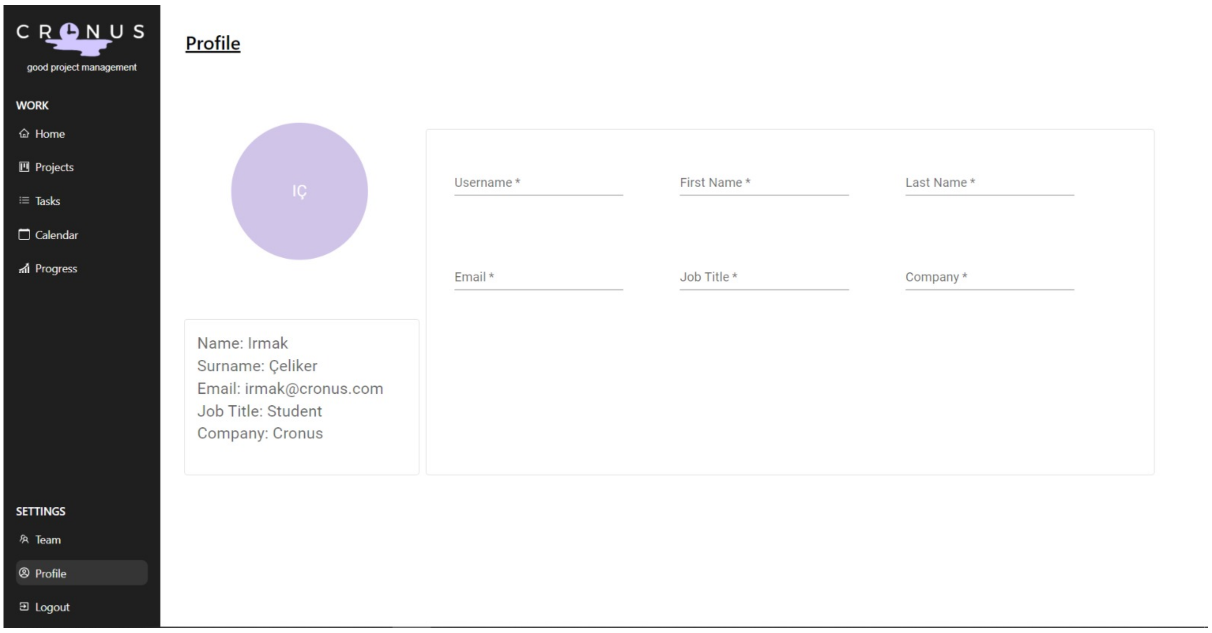


| Name | Lastname | Position | ProjectID |
|---|---|---|---|
| ProjectName: Cronus - 4 items | | | |
| Gizem | Karal | Backend Developer | 1 |
| Fatih | Karahan | Backend Developer | 1 |
| Gökberk | Boz | Frontend Developer | 1 |
| Irmak | Çeliker | Frontend Developer | 1 |
| ProjectName: Font - 2 items | | | |
| Elif | Arslan | Product Manager | 2 |
| Can | Senorman | Analyst | 2 |
| ProjectName: Prof - 2 items | | | |
| Gökberk | Boz | Backend Developer | 3 |
| Irmak | Çeliker | Backend Developer | 3 |

1 of 1 pages (8 items)

- In the team section, each project and their team members are shown.
- The distribution of the members and their assignments can be changed.

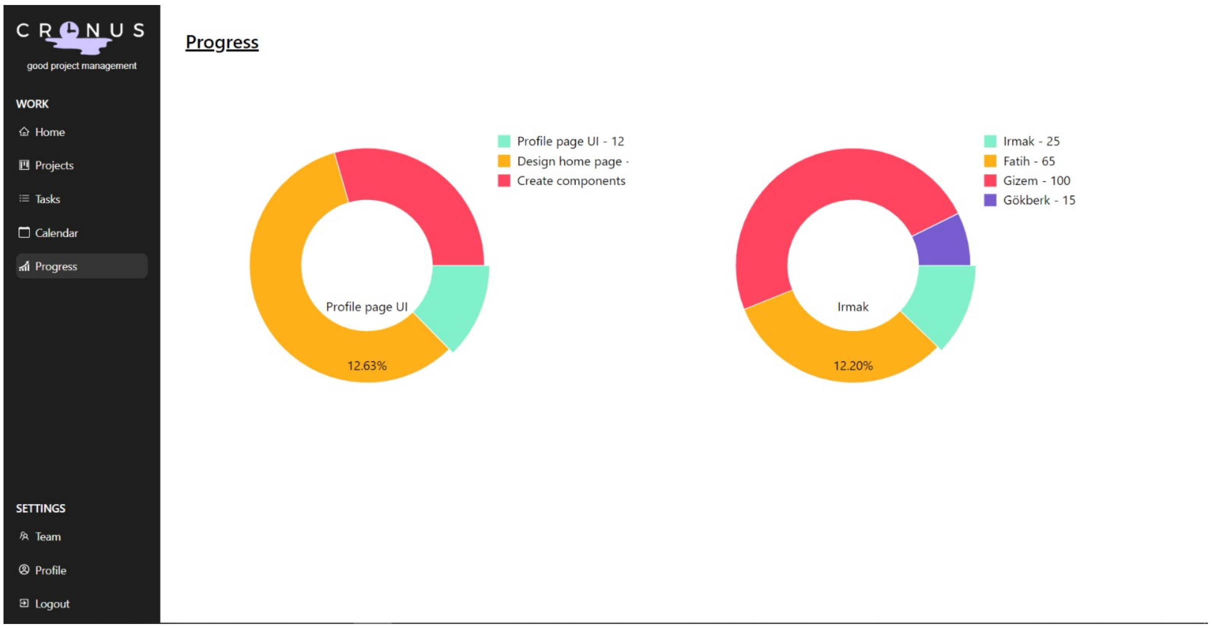
## Profile



Name: Irmak
Surname: Çeliker
Email: irmak@cronus.com
Job Title: Student
Company: Cronus

- In the profile page, the details about that person can be added or edited.

## Progress



- In the progress page, the user’s own progress is shown according to the projects he is a member of.

## 10. References

[1] React. “ *React*”. [Online]. Available: https://tr.reactjs.org/.  [Accessed: 24 Dec-2021].

[2] Next js. “ *Next js*”. [Online]. Available: https://nextjs.org/.  [Accessed: 24 Dec-2021].

[3]PostgreSQL. “ *PostgreSQL*”. [Online]. Available: https://www.postgresql.org/. [Accessed: 24 Dec-2021].

[4]Webhook. “ *What is a webhook: How they work and how to set them up*”. [Online]. Available: https://www.getvero.com/resources/webhooks//. [Accessed: 24 Dec -2021].

[5] Websocket. “ *The WebSocket API (WebSockets)*”. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API. [Accessed: 24 Dec- 20201].

[6] “9 Non-Functional Requirements”
Google. [Online]. Available:
https://www.indeed.com/career-advice/career-development/non-functional-requirements-examples