



Bilkent University Department of Computer Engineering

Senior Design Project

Project Name: Cronus

Analysis Report

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

Supervisor: İbrahim Körpeoğlu

Jury Members: Shervin Arashloo and Hamdi Dibeklioglu

Project Specifications Report Oct 11, 2021

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 CS491.

Table of Contents

1. Introduction	3
2. Proposed System	3
2.1. Overview	3
2.2 Functional Requirements	3
2.2.1. Client-side Web App	3
2.2.2. Backend Server	4
2.1.2. Admin Webapp	5
2.2 Non-functional Requirements	5
2.2.1 Usability	5
2.2.2 Reliability	5
2.2.2 Localization	6
2.2.4 Availability	6
2.2.5 Security	6
2.2.6 Performance	6
2.4 Pseudo Requirements	7
2.5 System Models	7
2.5.1 Scenarios	7
3.5.2 Use Case Model	11
3.5.3 Object and Class Model	11
3.5.4 Dynamic Models	11
3.5.5 User Interface - Navigational Paths and Screen Mock-ups	11
3. Other Analysis Elements	14
3.1 Consideration of Various Factors in Engineering Design	14
3.1.1 Public Health	14
3.1.2 Economic Factors	14
3.1.3 Sustainability Factors	14
3.1.4 Societal Factors	15
3.1.5 Cultural Factors	15
4.2 Risks and Alternatives	15
3.3 Project Plan	16
3.4 Ensuring Proper Teamwork	23
3.5 Ethics and Professional Responsibilities	23
3.6 Planning for New Knowledge and Learning Strategies	23
4. Glossary	23
5. References	23

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. Proposed System

2.1. Overview

Cronus is a web application that is designed for the project management process and for people who want to assign tasks and follow them. To be able to reach more users, the application will be free to use.

To make the reader have a better understanding of the application, it is thought by the team that it might be useful to explain some of the most important requirements in plain English sentences also, right along with upcoming sections. The application requires an account for the user to attend to the system.

After creating an account, the type of the user is determined, which can be the role manager, a participant in the team or admin. According to the type of the user, there are different actions permitted. The manager of the project opens a new project and adds members to the team of that project. Roles are given to each member which can be creating, deleting, editing issues; managing roles or managing the project. The role manager assigns roles and issues to the team members. With the help of the application, everyone can see who has accomplished their task. As a significant difference from other project management tools, our application helps to maintain the repetitive task management in the project. The user with the permission can change the role of the member, assign a new one or create a new assignment and determine the properties of the new role. The properties of the issue are due date, issue type, requirements or the period of time needed for it. Any issue may be deleted or renewed according to the plan by the manager. Every task completed is checked by who is assigned and it is seen by the other members in the team.

All the details on the application's requirements will be discussed in the next 4 sections. Later on, system models will be shown as visuals.

2.2 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 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.

2.4 Pseudo Requirements

- Application will be structured based on the Domain Driven Development method.
- The main pattern that will be followed during development is CQRS.
- For the implementation of the application we will be using C# for the backend, and Typescript for the frontend.
- We will be using Rider and Visual Studio Code as our IDEs.
- For the version control system we will be using Git, hosted on GitHub.
- For the backend web server implementation we will be using ASP.NET, with Entity Framework as the ORM and MediatR for the CQRS system.
- PostgreSQL will be used as the database for the application.
- For searching through the data we will be using Elastic Search.
- We will be using Seq as our log sink.
- For the frontend client we will be using a Next.JS application built with either Chakra UI, Material UI or TailwindCSS as the styling library.
- Web application will be following the a11y project standards for accessibility.
- Our deployment stack will be a Docker swarm. We may decide to move to kubernetes if need arises.
- Application will be hosted on a VPS instance. This decision may change as we near deployment.
- Any hosting expenses will be covered by the team.

2.5 System Models

2.5.1 Scenarios

Scenario 1 - Setup account

Actors: Customer

Entry Condition: User opens the website

Exit condition: User is navigated to the teams page

OR

User closes the app

- A customer subscribing to the service (*How does subscription work?*)
 - Subscriptions are handled on a monthly/yearly basis per user, handled by Stripe.
- Customer logging in to the service (*What are the ways to login?*)
 - Users can login via email and password.
- Creating a team, adding members to a team (*How do you add a user? Do you create their accounts, or do you invite them via their emails?*)
 - Users can be added by creating accounts for each user, either manually or by uploading a list of the users.
- Creating a project
- Creating a role for that team
- Editing the permissions of that role (*What are the different permissions?*)
 - There are multiple possible permissions that can be determined for each role:
 - Create/Edit/Delete issues
 - Manage Roles
 - Manage Projects
- Adding members to the different roles (*What determines who can add what roles?*)

- Only a user who has a role with *Manage Roles* permission can add roles to other users, but only roles that have higher permissions.
- Creating a task
- Assigning members to the task

Scenario 2 - Create Issue

Actors: Users with appropriate permissions

Entry condition: User clicks the create issue button

Exit condition: User clicks the create button

OR

User closes the app

- A team manager logging in
- Viewing all the teams they're part of (*What are the ways to view the teams?*)
 - Teams can be listed by their member count, issue count, with the ones that have most issues closest to their due dates, etc.
- Viewing the projects of the team (*What are the ways to view the projects?*)
 - Same options for viewing the teams applies to the projects as well
- Opening a project (*What is on the project page?*)
 - On a project page a user can see the list of issues divided by their repeat periods, as well as the name and the description of the project
- Creating an issue - *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*
- Adding a period to the issue (*What are the possible periods?*)
 - Anything that repeats on at most a yearly basis can be a period.

- Adding requirements to an issue (*What is a requirement?*)
 - A requirement is a prerequisite issue that needs to be completed before the issue can be completed.
- Giving the connected issues a name (*What else can be done with connected issues?*)
 - Connected issues can be edited in bulk, such as adding users, changing periods or disabling.
- Adding users to an issue (*Who can you add?*)
 - Only team members that this project belongs to can be added to an issue.
- Changing the issue type (*What are the issue types?*)
 - An issue can be *collective* or *cumulative*, depending on whether the issue needs to be completed by a single person assigned or by everyone assigned.
- Adding a deadline to the issue (*How is a deadline determined?*)
 - A deadline is a time period (such as 3 days) that the issue needs to be completed in.

Scenario 3 - Disable Issue

Actors: Users with appropriate permissions

Entry condition: User clicks the disable issue button

Exit condition: User approves disabling the issue

OR

User closes the app

- A team manager logging in
- Viewing all the teams they're part of
- Viewing the projects of the team
- Opening a project
- Disabling an issue (*How does disabling work?*)

- A project can be disabled by a team manager that has the appropriate permissions for X number of periods.

Scenario 4 - Complete issue

Actors: Users with appropriate permissions

Entry condition: User clicks the complete issue button

Exit condition: User closes the app

- A team member logging in
- Viewing all the issues assigned to them.
- Tagging the completed issue as done

Scenario 5 - Extend issue

Actors: Users with appropriate permissions

Entry condition: User clicks extension button

Exit condition: User submits the extension form

OR

User closes the app

- A team member logging in
- Viewing all the issues assigned to them.
- Tagging the problematic issue as needing extension (*How does asking for extension work?*)
 - A member who is assigned to the issue can mark the issue for needing extension, with an optional reason and optional extension amount.

Scenario 6 - View extension notification

Actors: Users with appropriate permissions

Entry condition: User clicks the notifications icon

Exit condition: User clicks the back button

OR

User closes the app

- A team manager logging in

- Viewing all the notifications from the teams they're managing
- Opening the issue that was flagged as needing extension
- Reading the reason for the extension
- Extending the deadline

Scenario 7 - Searching a past issue

Actors: User

Entry condition: User navigates to the search page

Exit condition: User clicks the back button

OR

User closes the app

- A team member logging in
- Opening the search menu
- Picking the search type as an issue (*What are the other search types?*)
 - Teams, team members and projects are the other search types.
- Filtering the issues only for completed ones (*What are the other filtering options?*)
 - Any field of an issue can be used for filtering
- Viewing the results

Scenario 8 - Viewing the logs

Actors: Users with appropriate permissions

Entry condition: User navigates to the logs page

Exit condition: User clicks the back button

OR

User closes the app

- A team manager logging in
- Opening the logs page
- Filtering the logs for issues that was being declared done after the due date

- Viewing which team members were the ones not being able to meet their deadlines

Scenario 9 - Fixing the wrongly assigned user

Actors: Users with appropriate permissions

Entry condition: User opens the issue page

Exit condition: User clicks the back button

OR

User closes the app

- A team manager logging in
- Finding the issue that had someone assigned mistakenly
- Removing the user from the issue
- Adding the correct user(s) to the issue

Scenario 10 - Remove user

Actors: Users with appropriate permissions

Entry condition: User navigates to the users page

Exit condition: User clicks the back button

OR

User closes the app

- An admin logging in
- Viewing the list of users connected with their account
- Searching for the user they want
- Removing the user from the account

```

graph TD
    subgraph Manager
        Manager((Manager))
        SignUp([Sign up])
        SignIn([Sign In])
        CreateTeam([Create a Team])
        CreateProject([Create a Project])
        EditPermission([Edit Permission])
        ViewingLogs([Viewing Logs])
        ManageProjects([Manage Projects])
    end

    subgraph NormalMember
        NormalMember((Normal Member))
        ViewExt([View Extension])
        SearchPastIssue([Search Past Issue])
        SeeLogDates([See Log Dates])
        CreateIssue([Create Issue])
        ManageRoles([Manage Roles])
        CompleteIssue([Complete Issue])
        DisableIssue([Disable Issue])
    end

    Manager --> SignUp
    Manager --> SignIn
    Manager --> CreateTeam
    Manager --> CreateProject
    Manager --> EditPermission
    Manager --> ViewingLogs

    NormalMember --> ViewExt
    NormalMember --> SearchPastIssue
    NormalMember --> SeeLogDates
    NormalMember --> CompleteIssue
    NormalMember --> DisableIssue

    SignUp -.->|<<Include>>| SignIn
    SignIn -.->|<<Extend>>| RemoveUser([Remove User])
    CreateTeam -.->|<<Extend>>| CreateRole([Create Role of team])
    CreateTeam -.->|<<Extend>>| AddMembers([Add Members])
    EditPermission -.->|<<Include>>| CreateIssue
    EditPermission -.->|<<Include>>| ManageRoles
    EditPermission -.->|<<Include>>| ManageProjects
    SeeLogDates -.->|<<Include>>| CreateIssue
    CreateIssue -.->|<<Include>>| ManageRoles
    CompleteIssue -.->|<<Include>>| DisableIssue
  
```

Figure 1: The Main Use Case Diagram of Cronus

3.5.3 Object and Class Model

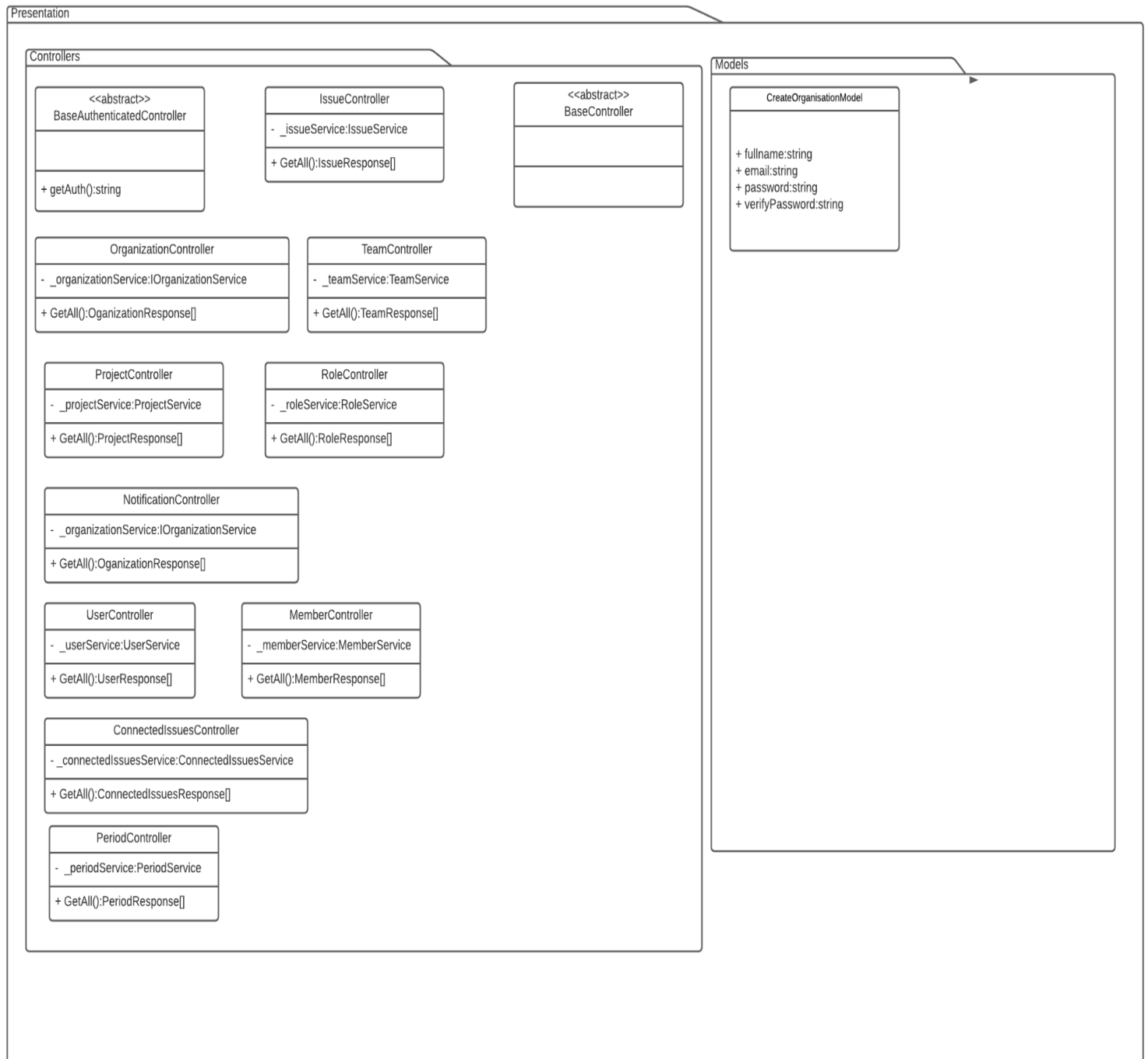


Figure 2: Object Model Presentation of Cronus

Figure 3: Domain Class Diagram of Cronus

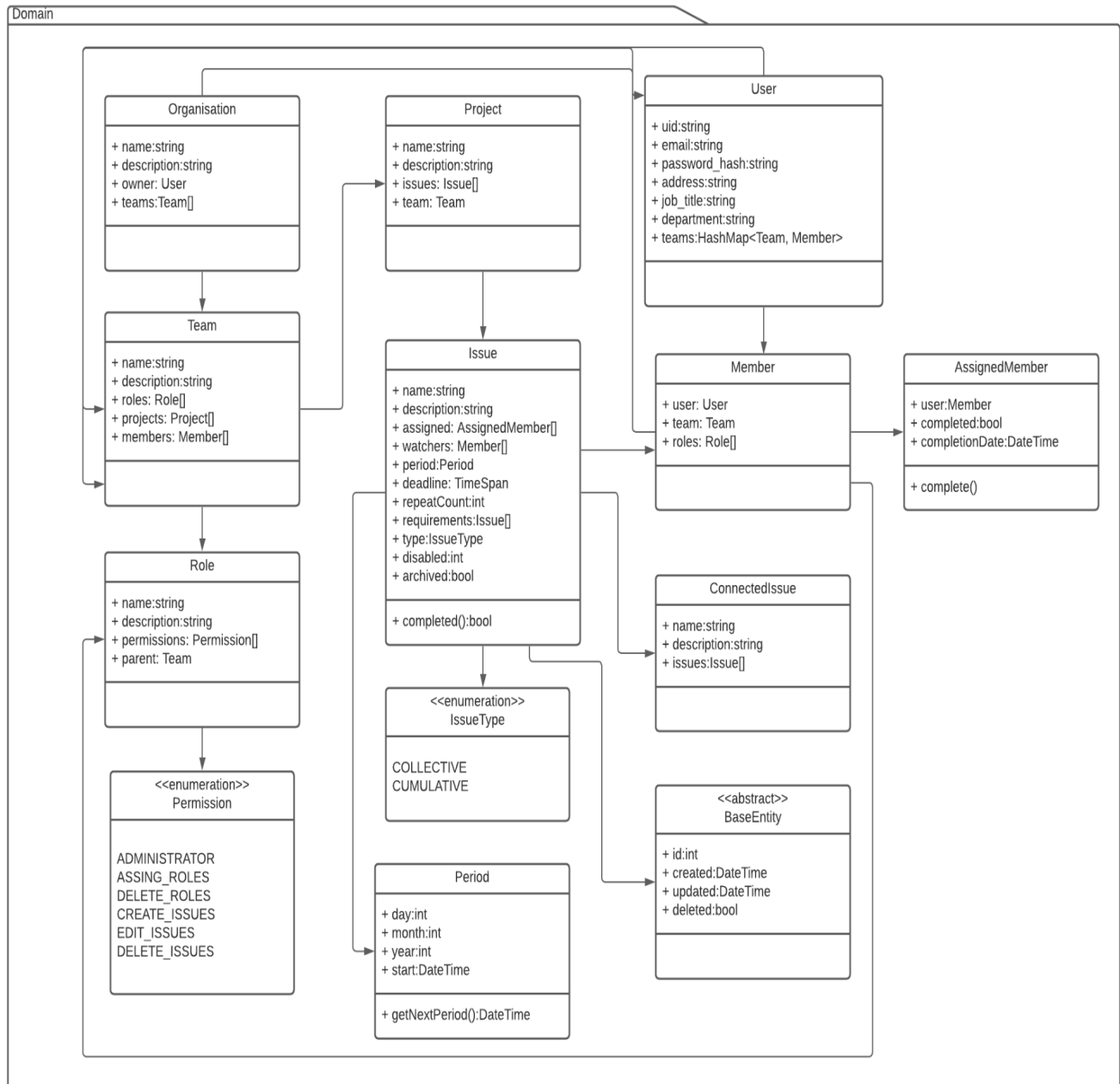


Figure 4: Application Object Diagram

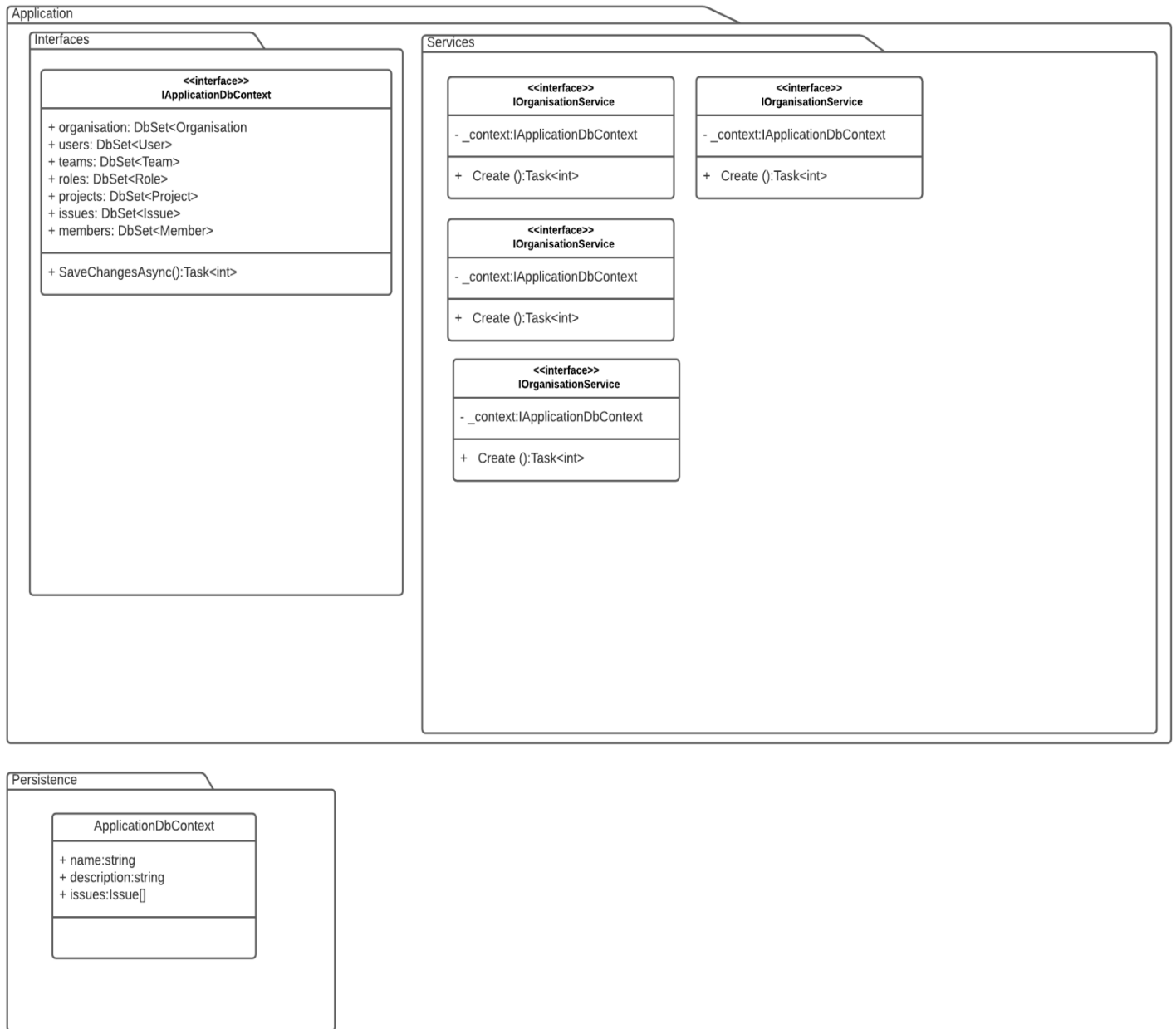
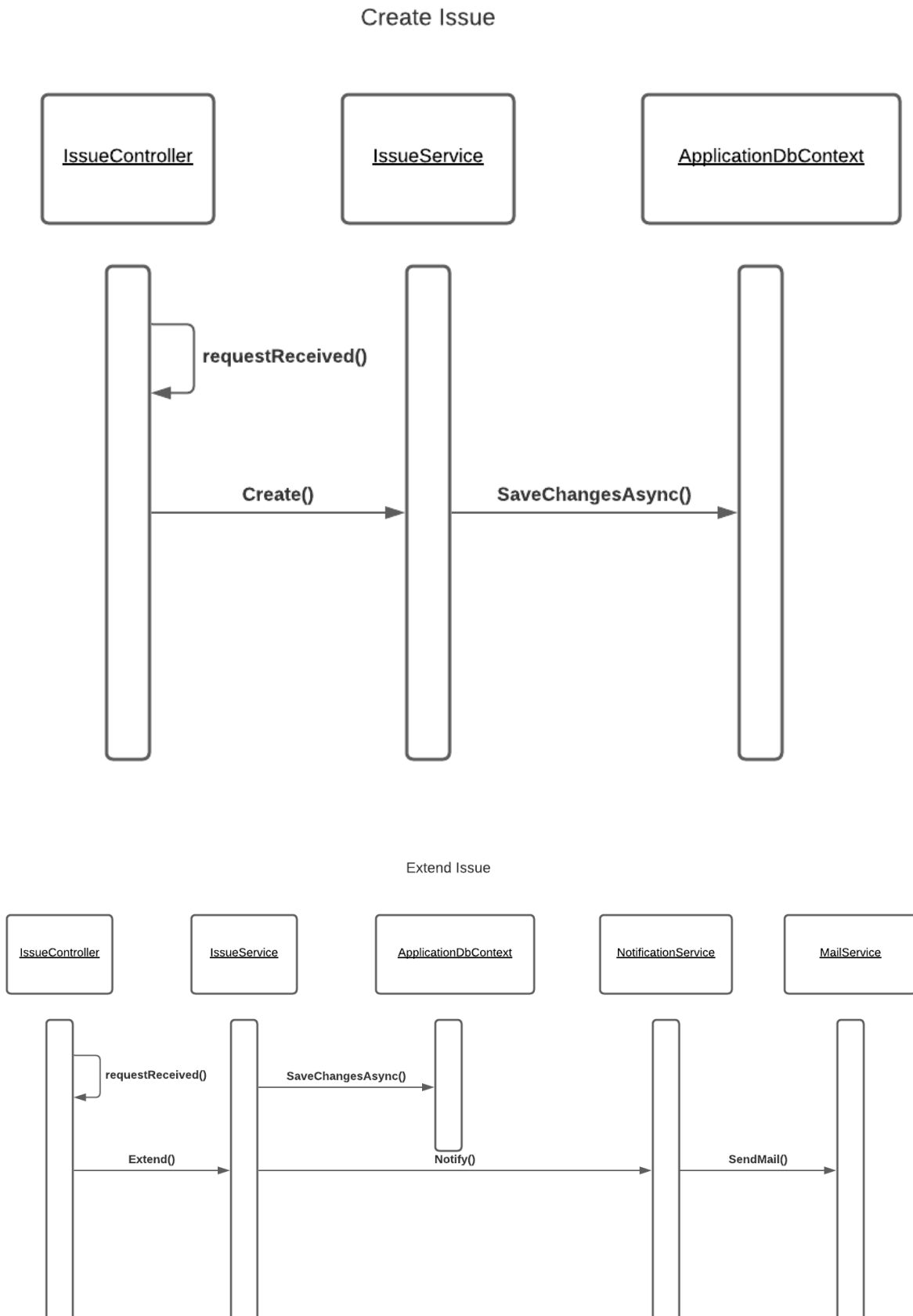
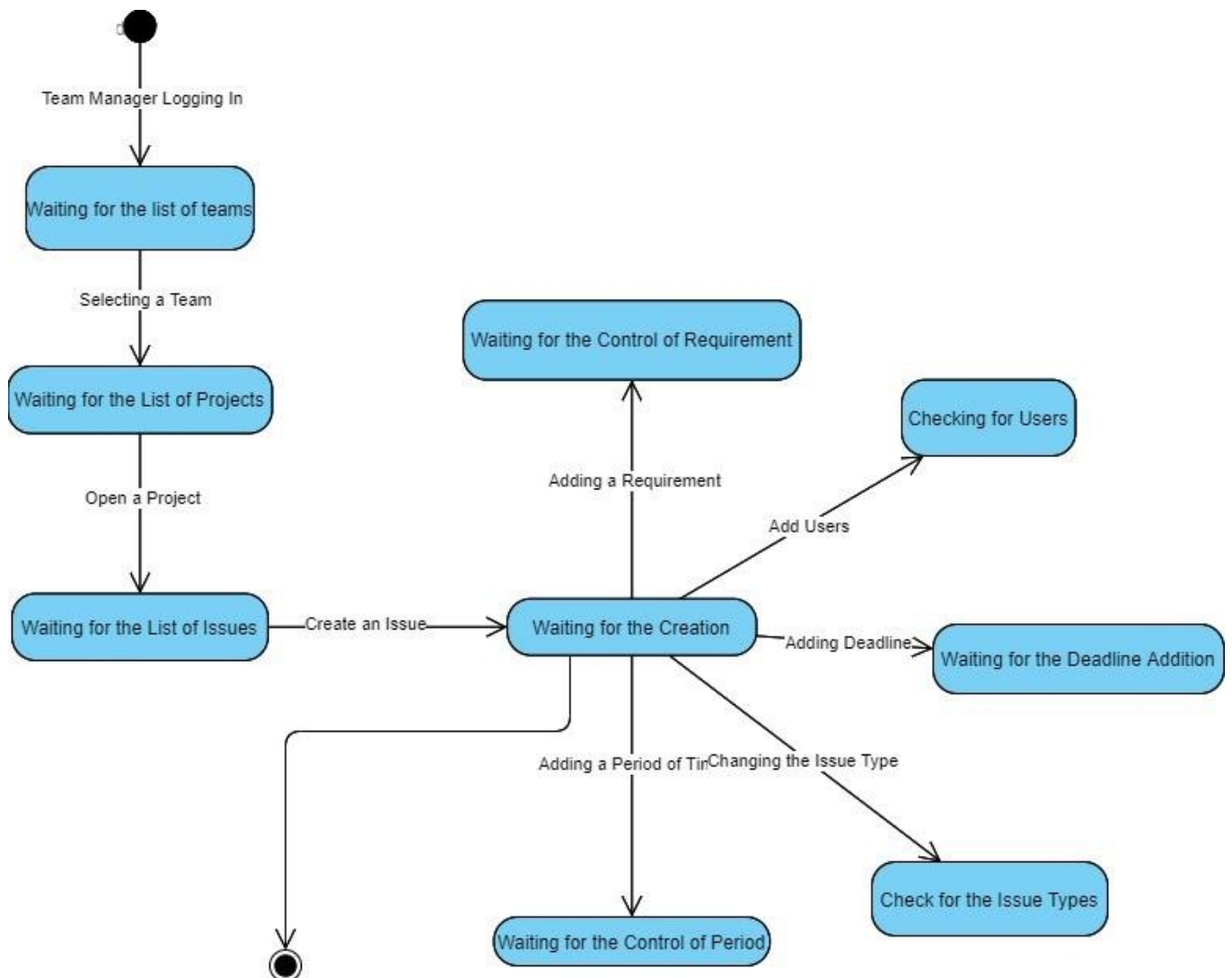


Figure 5: Create and Extend Issue Sequence Diagram



3.5.4 Dynamic Models

Figure 6: State diagram of user



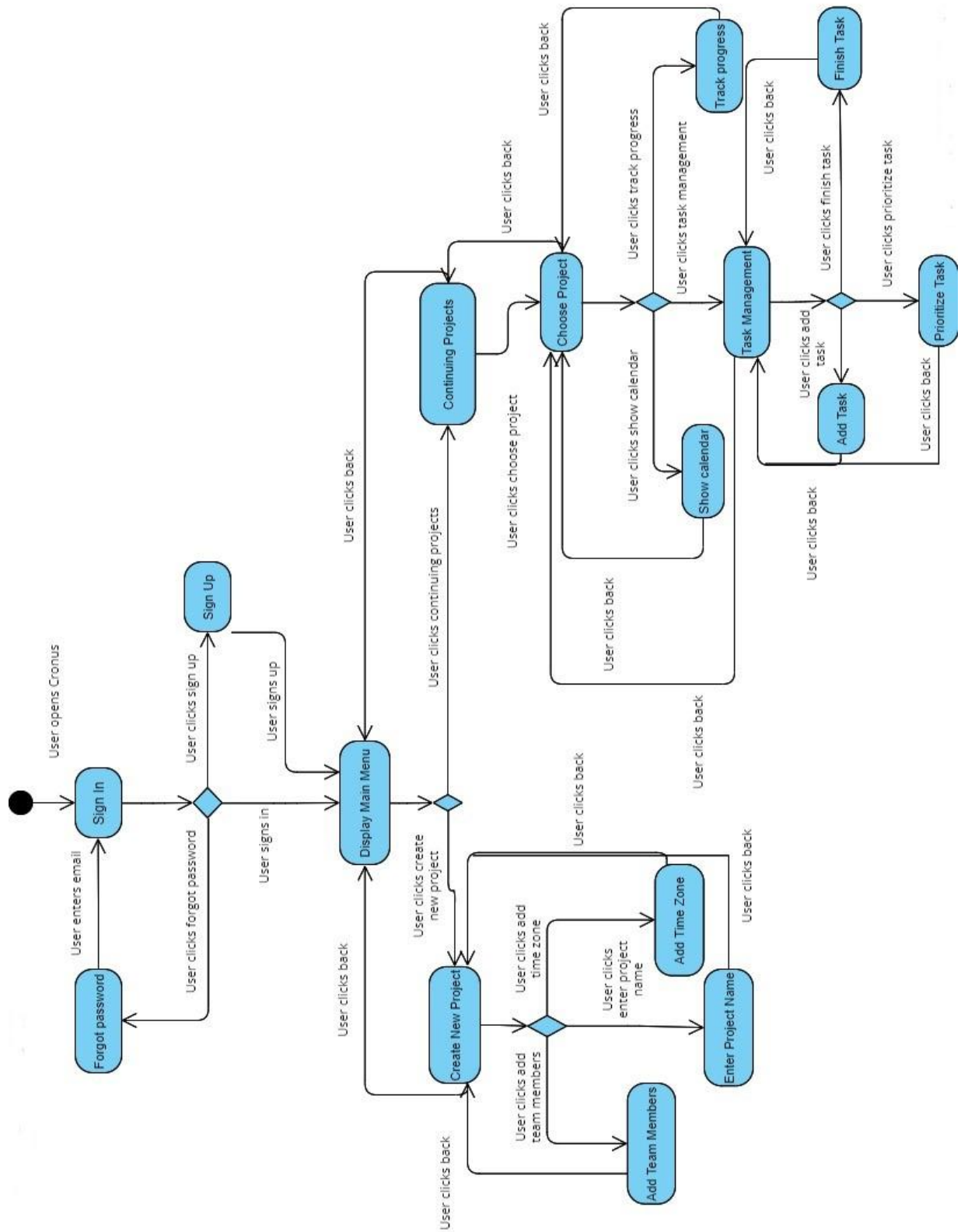


Figure 7: Active Diagram of User

3.5.5 User Interface - Navigational Paths and Screen Mock-ups

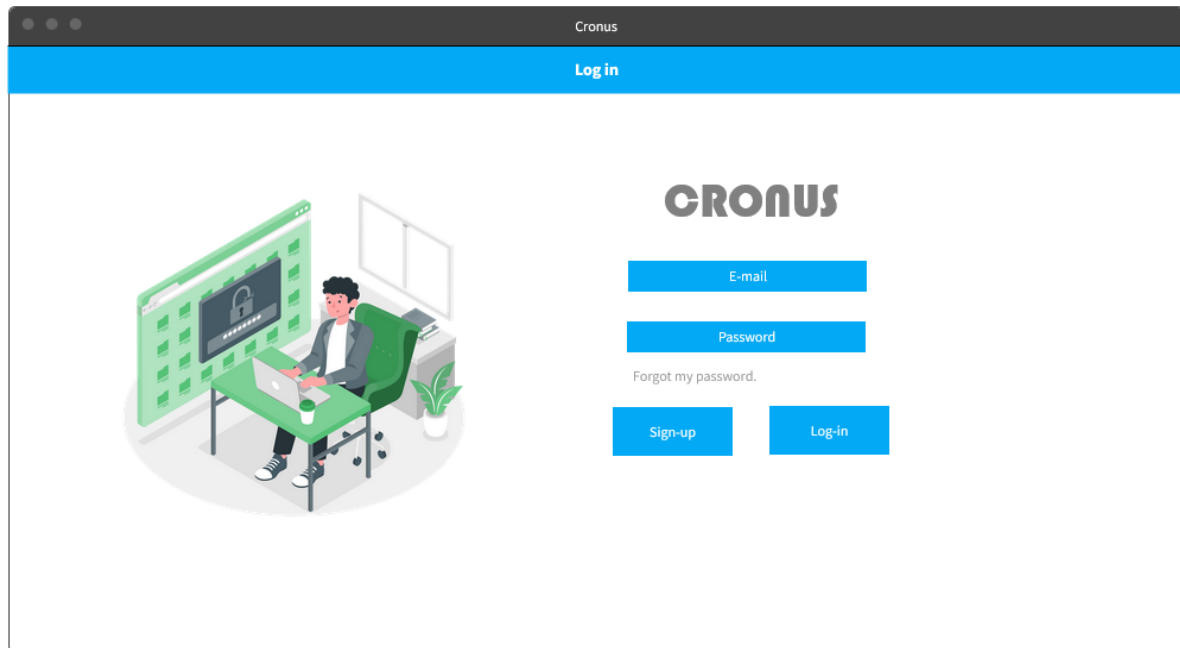


Figure 8: Login Screen

Cronus						
Your/Your Company's Workspace			Week	Month	Colleagues	Options
			9.11.2021			
	08.00-10.00	10.00-12.00	12.00-14.00	14.00-16.00	16.00-18.00	
Monday	Company daily reports given.	Weekly plan meeting			Data share with Coworkers	
Tuesday	Company daily reports given.			Intern meeting		
Wednesday	Company daily reports given.		Mid-week Meeting			
Thursday	Company daily reports given.	Meeting with other team		Intern meeting		
Friday	Company daily reports given.			Weekly report	Happy hour	

Figure 9: Weekly Schedule of a Person

Cronus

Your/Your Company's Workspace

Week Month Colleagues Options

Issue name...

Days

☒ Weekly

☐ Monthly

Issue explanation..

[Invite Colleagues](#) ✓

Figure 10: Creating an Issue

Cronus

Your/Your Company's Workspace

Week Month Colleagues Options

Team	Company	Issues
Alpha Team	Company A	2 Issues for this week
Charlie Team	Company A	➔
Delta Team	Company A	3 Issues for this week
Echo Team	Company A	6 Issues for this week
Fox Team	Company A	No issue for this week

Company Daily Report

Intern Meeting

Mid-week Meeting

Weekly Report

Happy Hour

Figure 11: Organization Leader Team Review

Cronus

Your/Your Company's Workspace

Week Month Colleagues Options

Search by name, team or company...


Name	Team	Company	Actions
Janice Monahan	Alpha Team	Company A	
Rollin Fadel	Alpha Team	Company A	
Lera Stroman	Wolfes	Company D	
Adan Schiller	Charlie Team	Company A	
Tony Brown	Redx	Company X	
Michael Liter	Eagles	Company D	

Figure 12: Colleagues and Actions

Cronus

Your/Your Company's Workspace

Week Month Colleagues Options



Project name...

Select the teams

Project explanation...

Calendar for creating issues.

☒ Include 2 regular weekly meeting.
 ☐ Invite all the company.
 ☒ Repeat each month.

< November 2021 >

Mo	Tu	We	Th	Fr	Sa	Su
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					

Figure 13: Creating a project.

3. Other Analysis Elements

3.1 Consideration of Various Factors in Engineering Design

3.1.1 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.

3.1.2 Economic Factors

Cronus will not require a payment. Every user will be able to access the same features and everything will be free. However, it aims to speed up the project management process, so the application can help individuals or companies to profit indirectly and the employment rate may increase.

3.1.3 Environmental Factors

There are not any environmental factors to be considered.

3.1.4 Societal Factors

By easing up the process of project management, Cronus aims to reduce 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.

3.1.5 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.

Table 1: Factors to consider during the design.

	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

4.2 Risks and Alternatives

Cronus aims to help the project management process of teams by letting team members prioritize tasks, creating calendars, track each other's tasks and progress. All the team members can add tasks to the calendars and prioritize them, however team members can have different opinions and preferences during these processes and if they all have the same authority in the project this may cause a problem. The main goal behind this application is to reduce the conflicts and chaos that can happen between the team members, since there won't be a chatting option, a plan B is needed for this situation. An extra feature where team members can vote on prioritizing and adding the tasks will be added in order to avoid these kinds of complications.

Cronus aims to bring people together from all around the world, so the differences in time zones may create a problem during creating a calendar. To avoid this complication, during the creation of every project a specific time zone can be

selected before starting, so that every team member can arrange themselves according to that time zone.

Table 2: Risks

	Likelihood	Risk effect on the project	B Plan
Different preferences in task management	6	Risk of conflicts between team members that can remain unresolved due to the lack of communication	Extra option to vote between team members before prioritizing and adding tasks
Team members in different time zones	8	Risk of confusion in terms of time arrangement between team members	Option to choose a common time zone for the project

3.3 Project Plan

WP 1: Analysis			
Start date: 20 September 2021 End date: 13 November 2021			
Leader:	Gizem Karal	Members Involved:	Irmak Çeliker Gökberk Boz Farih Karahan
<p>Objectives: The purpose of this work package is to explore the requirements and other factors that should be taken into account when designing the project.</p> <p>Tasks:</p> <p>Task 1.1 Requirement Meetings: Analyzing and determining all requirements and specifications of the project.</p> <p>Task 1.2 Specification Report: Writing a report about the specifications and requirements of the project.</p> <p>Task 1.3 Analysis Report: Writing a report about the use cases, object and class models as client class diagram, server side class diagram, and dynamic models. Also, determining the user interface design.</p> <p>Deliverables</p> <p>D1.1: Specification Report</p> <p>D1.2: Analysis Report</p>			

Table 3: Work Schedule 1

WP 2: Design			
Start date: 20 November 2021 End date: 21 December 2021			
Leader:	Irmak Çeliker	Members Involved:	Gizem Karal Gökberk Boz Farih Karahan
<p>Objectives: The purpose of this work packaga is to explore the requirements and other factors that should be taken into account when designing the project. This part consists of high and low-level designs of the project.</p>			
<p>Tasks:</p> <p>Task 2.1 Evaluation of the Analysis: Analyzing the design of the project according to the functional and nun-functional requirementsby determining the scenarios.</p> <p>Task 2.2 High-Level Design: Determining and designing the project design goals and dividing the project into subsystems.</p> <p>Task 2.3 Low-Level Design: Defining design principles to be used in the project.</p>			
<p>Deliverables</p> <p>D2.1: High-Level Design Report</p> <p>D2.2: Low-Level Design Report</p>			

Table 4: Work Schedule 2

WP 3: Development 1			
Start date: 20 October 2021 End date: 27 December 2021			
Leader:	Gökberk Boz	Members Involved:	Gizem Karal Irmak Çeliker Farih Karahan
<p>Objectives: The purpose of this study module is to process and prepare the general and skeleton structure of the project. Also, getting ready for the demonstration of the project and presentation of it.</p>			
<p>Tasks:</p> <p>Task 3.1 Creation of Database: Deciding which database will be used.</p> <p>Task 3.2 Implementation of User Interface: Implementing the user interface of Cronus with the help of our design report.</p> <p>Task 3.3 Implementation of Some Features: Implementating some features for the first demonstration.</p>			
<p>Deliverables</p> <p>D3.1: Demo and First Presentation</p>			

Table 5: Work Schedule 3

WP 4: Development 2			
Start date: 1 January 2022 End date: 13 April 2022			
Leader:	Fatih Karahan	Members Involved:	Gizem Karal Gökberk Boz Irmak Çeliker
<p>Objectives: The purpose of this study module is to implement the advanced level of project structure and getting ready for the second demonstration and presentation.</p>			
<p>Tasks:</p> <p>Task 4.1 Repetitive Tasks Management System: Implementing the management system for repetitive tasks.</p> <p>Task 4.2 Server and Database: Implementing the server side and database according to the specification and requirements.</p> <p>Task 4.3 Implementation of the Remaining Features: Defining design principles to be used in the project.</p>			
<p>Deliverables</p> <p>D4.1: Repetitive Tasks Management System</p> <p>D4.2: Server and Database</p>			

Table 6: Work Schedule 4

WP 5: Testing			
Start date: 3 December 2021 End date: 5 May 2022			
Leader:	Gizem Karal	Members Involved:	Irmak Çeliker Gökberk Boz Farih Karahan
<p>Objectives: The purpose of this study module is to test the succeeded and implemented parts of the project. After this package, verification and validation of the project will be accomplished.</p>			
<p>Tasks:</p> <p>Task 5.1 Repetative Tasks Management System Testing : Testing this system implemented from the previous study module.</p> <p>Task 5.2 Server and Database Testing: Testing connection between the application and server and database.</p> <p>Task 5.3 Application Testing: Testing the whole projects with its every features implemented.</p>			
<p>Deliverables</p> <p>D5.1: Cronus Application</p>			

Table 7: Work Schedule 6

3.4 Ensuring Proper Teamwork

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.

3.5 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.

3.6 Planning for New Knowledge and Learning Strategies

Our main consideration is to make everyone study similar materials during the development. During the project, we will learn a new concept of web development which will focus on project management. So many resources are available on the internet, however we should go through similar resources in order to maintain the order and accuracy in the project. Everyone considering a different resource may cause distraction and waste of time. All new concepts will be comprehended by self learning by searching on the internet. We will watch high level web development videos and look for documentation. At this point, it is quite significant for everyone to improve their self-learning.

4. References

[1] "Coronavirus Disease(COVID-19)," Google. [Online]. Available: <https://www.who.int/emergencies/diseases/novel-coronavirus-2019>. [Accessed: 11-Oct-2021].

[2] "What is Kanban?," Google. [Online]. Available: <https://www.digite.com/kanban/what-is-kanban/>. [Accessed: 11-Oct-2021].

[3] "Jira Software," Google. [Online]. Available: <https://www.atlassian.com/software/jira/guides/getting-started/overview>. [Accessed: 11-Oct-2021].

[4] "Asana," Google. [Online]. Available: <https://asana.com/>. [Accessed: 11-Oct-2021].