FH KIEL TICKETING APP

Kriti Agarwal - 930104

Mohammad Hussain Hussaini - 929876

Muhammad Adnan - 929908

Pooja Murarka - 930090

Table of Contents

1 Introduction

1.1 Purpose

This document contains a detailed description of the FH Kiel Ticketing App. The document contains an overall description of the product, the scope of the application, the intended audience, the functional and the non-functional requirements of the product. The document gives the reader a complete understanding of what the system will offer and all the features that the system will fulfill in its complete working condition.

What usually happens is that when a student wants to complete their project/thesis they must go to a professor and present them a proposal for an idea they have. They also sometimes go to different professors to talk about any ideas the professors might already have so that they (the students) can work on. Most of the time the students have no idea in what form the proposal is supposed to be in and what information must be included in a proposal. After all this has been taken care of, the whole process is carried out on emails and frequent meetings. This usually means a lot of workload on the professor since there would be a huge amount of emails to keep track of. In a nutshell, there is no organized structure that would benefit both students and professors at the moment.

The FH Kiel Ticketing App is a platform where the students can log in, fill in a certain form and raise a ticket so that the student can begin working on a Project or Thesis. The ticket is then reviewed by a professor who will decide whether or not to supervise the received project/thesis proposal. The FH Kiel Ticketing App caters for the students of Information Engineering in the Master's program of Fachhochschule Kiel initially. However the design of the software is scalable and can cater to multiple programs present in the university.

1.2 Document Conventions

Following are the conventions followed in the document:

1. Any word in **bold** form means that it has a different definition in the context of the document. These words are all nouns. The definition can be found in Appendix A: Glossary.
2. To avoid ambiguity, the words that are in *italic and have a superscripted alphabet* [x] (example) are defined in Appendix A: Glossary for a clearer context as to what the word means in the given circumstance. The reason for choosing an alphabet is that the document shall not have much content for context clarification that would exceed the alphabet count. Apart from that, a number inside square brackets is used to denote references and bibliography.
3. The document is divided into different parts and the headings are numbered. The numbering of the headings follow the following convention:
    a. 1. Main Heading
        i. 1.1 Sub-Heading of Heading 1
        ii. 1.2 Second Sub-Heading of Heading 1
            1. 1.2.1 Sub-Heading of Heading 1.2
    b. 2. Second Main Heading
        i. 2.1 Sub-Heading of Heading 1

1.3 Intended Audience and Reading Suggestions

This document is to be *mainly* [a] read by the following people:

1. The team members (Names given in the title page)
2. Project Owner
3. The FH Kiel Examination Department

The document is written according to the standard IEEE template for a Software Requirement Specification. The proper way of going through it would be in a chronological order. With every increasing heading point the document uncovers details that are important to understand the complete functionality. It is also suggested that when faced with confusion, the words that are either bold or followed by a superscripted number can be found in Appendix A: Glossary for a clearer understanding of its meaning and purpose.

1.4 Product Scope

The FH Kiel Ticketing App is made for students and professors so that they can have a platform where their master projects and thesis can be carried out in an organized and transparent manner. Since it is already discussed in the beginning that the whole process, currently, for getting starting with a project or thesis is confusing and difficult: The FH Kiel Ticketing App allows the students to either browse from already existing ideas that they have or they can form a proposal of an idea they already have. To make the proposal they don't need to know anything about what format it has to be in because the application would ask for the required data and generate a proposal for them. After generating this proposal a ticket will be raised which will take the student's proposal to a valid professor. The professor would then evaluate whether or not he/she would like to supervise the project/thesis. Upon evaluation the professor can either accept it, reject it or ask the student to make some changes. Once accepted the application will allow the professor and the student to exchange messages, files and view all of it in a mini-dashboard where they (the professor and the student(s)) know what exactly is going on i.e. complete transparency.

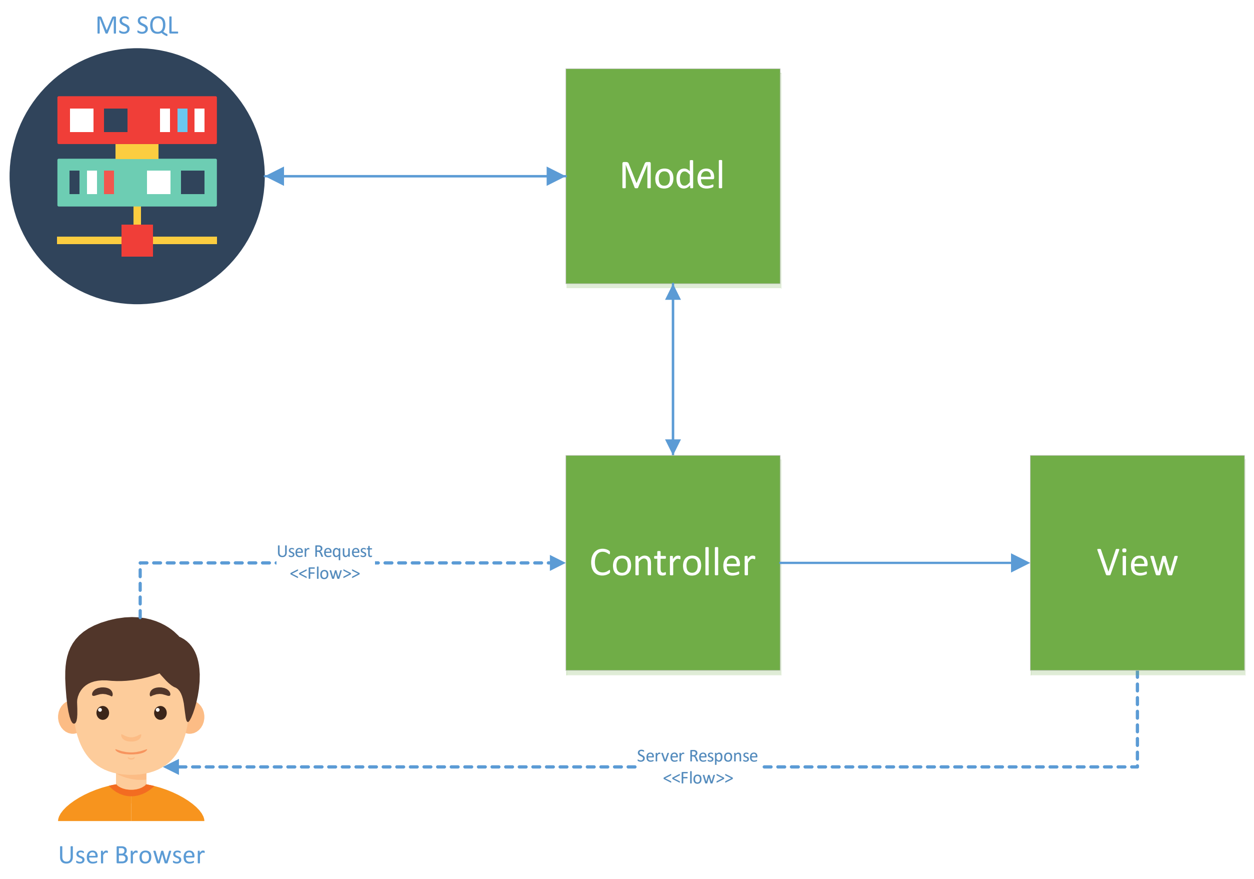
In regards to the information listed above, it is to be noted that this is the main business scope of the application. There are many detailed aspects of the product that are discussed further ahead.

2 Overall Description

2.1 Product Perspective

The FH Kiel Ticketing App is a standalone software. Even though Fachhochschule Kiel already has multiple applications currently running, for now, the FH Kiel Ticketing App will be standalone and operate without any integration to the current existing platforms of Fachhochschule Kiel. This product will be self-contained and standalone.

An overall description of the system can be found in the following component diagram. Based on the ASP MVC architecture it can safely be said that the product will have models, views and controllers that will communicate with each other and a MS SQL Database in the backend for data storage.



2.2 Product Functions

The FH Kiel Ticketing App will provide a lot of features that are defined and listed in detail further ahead in the document. However, the application will have the following main functions/features:

- Students have the ability to carry out their Master Projects through the application.
- Students have the ability to carry out their Master Thesis through the application.
- Students have the ability to create **Ideas** regarding their Projects/Thesis.
- Professors have the ability to create **Ideas.**
- Students have the ability to join available **Ideas.**
- Students have the ability to raise **Tickets** from the created **Ideas.**
- Professors have the ability to either accept or decline a **Ticket.**
- Professors have the ability to add another Supervisor in the **Ticket.**

2.3 User Classes and Characteristics

The following user classes are bound to use the FH Kiel Ticketing App.

- Student: The student will be the user who would initiate the flow of the process. The students will use the platform to engage in completing their Master Project/Thesis.
- Supervisor: The Supervisor will be a user that would use this platform so that they can have an organized location to handle their communication with the students they are working with. The supervisors are the Professors of FH Kiel.

- Admin: This is a special type of user and will be considered not a part of the topic of **user** later on in the document. Given that, it is a user class and has a certain role in the system. Initially when the system will be set up for deployment, the admin user will be the one who will set up the **roles, fields** and **role identifiers** in the system. Apart from the stated functionalities there might be other configuration actions that the admin must take which will also be a part of the final product. Other functionalities such as viewing graphs for number of tickets with their respective statuses, number of proposals rejected and number of students currently going through a working **ticket** will also be available to the admin user.

Both the user classes (student and supervisor) are important. They are the main drivers of the platform and the whole product revolves around fulfilling the needs of the students and supervisors.

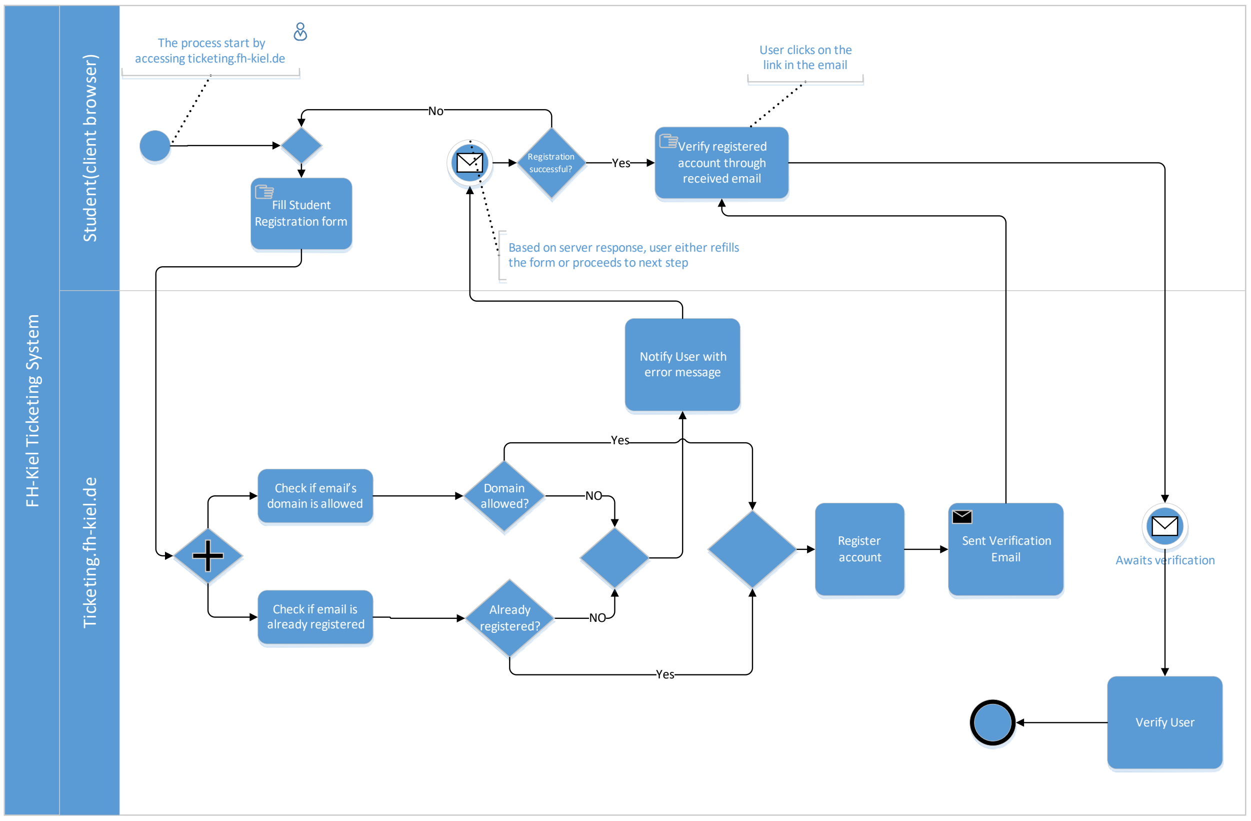
The above stated user classes are the two main users that will be using the product. However, there is a possibility that the following user class may enter in the scope and benefit from the platform *in the future*. [b]

2.4 Product Function – Business Process Models

To provide more insight to the functionality of the product, following are business process models that informs about the flow of the system. These business processes show a step by step illustration on how a process is carried out in the system by certain **users**.

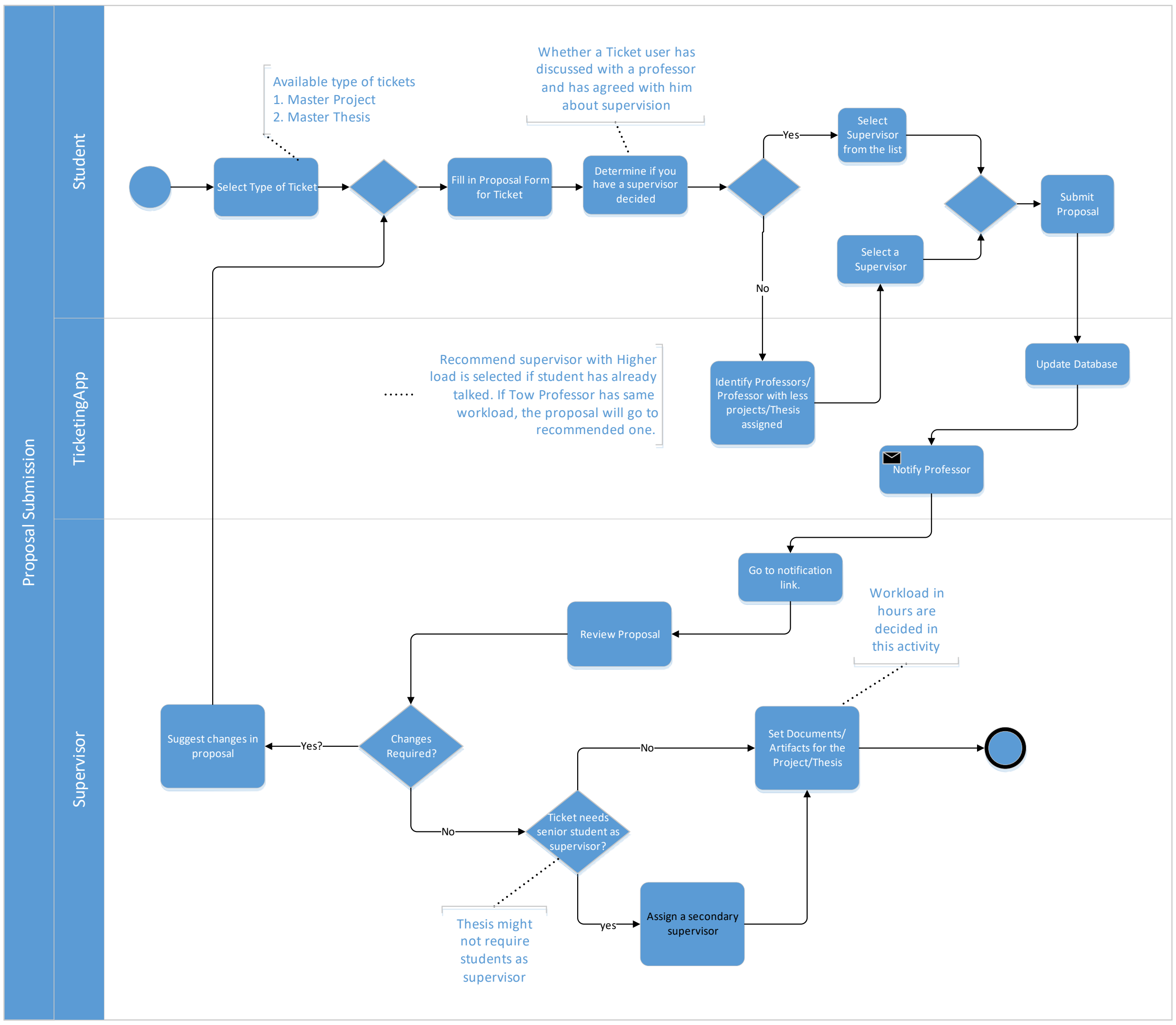
1. User Registration:

The following business flow diagram shows the flow of how a **user** registers in the system.


FH-Kiel Ticketing System
Student(client browser)
Ticketing.fh-kiel.de
The process start by accessing ticketing.fh-kiel.de
Fill Student Registration form
No
Registration successful?
Yes
Verify registered account through received email
User clicks on the link in the email
Based on server response, user either refills the form or proceeds to next step
Notify User with error message
Check if email's domain is allowed
Check if email is already registered
Domain allowed?
Already registered?
Yes
NO
NO
Yes
Register account
Sent Verification Email
Awaits verification
Verify User

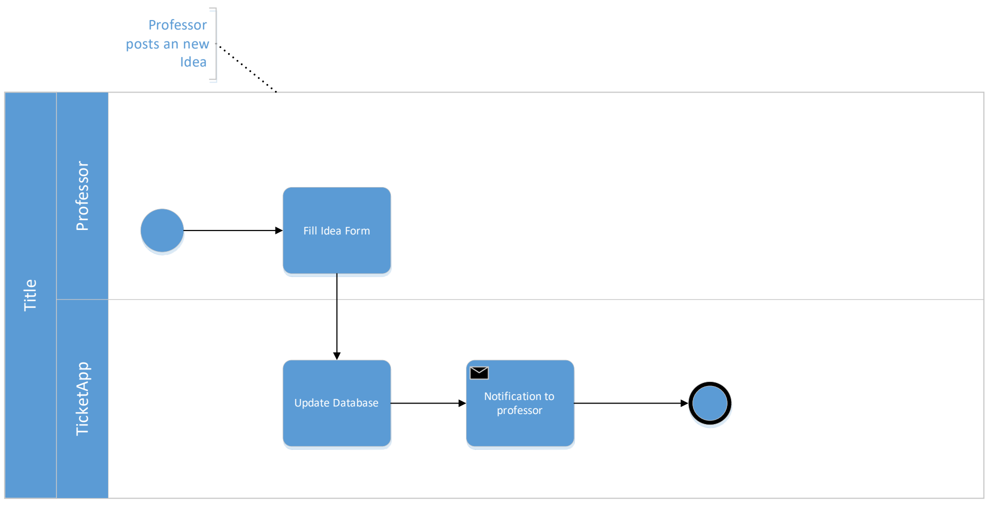
2. Proposal Submission

The following business process model shows how a proposal is submitted in the system by a student.


Proposal Submission
Student
TicketingApp
Supervisor
Available type of tickets
1. Master Project
2. Master Thesis
Select Type of Ticket
Fill in Proposal Form for Ticket
Whether a Ticket user has discussed with a professor and has agreed with him about supervision
Determine if you have a supervisor decided
Yes
Select Supervisor from the list
No
Select a Supervisor
Submit Proposal
Recommend supervisor with Higher load is selected if student has already talked. If Tow Professor has same workload, the proposal will go to recommended one.
Identify Professors/ Professor with less projects/Thesis assigned
Update Database
Notify Professor
Go to notification link.
Workload in hours are decided in this activity
Review Proposal
Suggest changes in proposal
Yes?
Changes Required?
No
Ticket needs senior student as supervisor?
No
Set Documents/ Artifacts for the Project/Thesis
yes
Assign a secondary supervisor
Thesis might not require students as supervisor

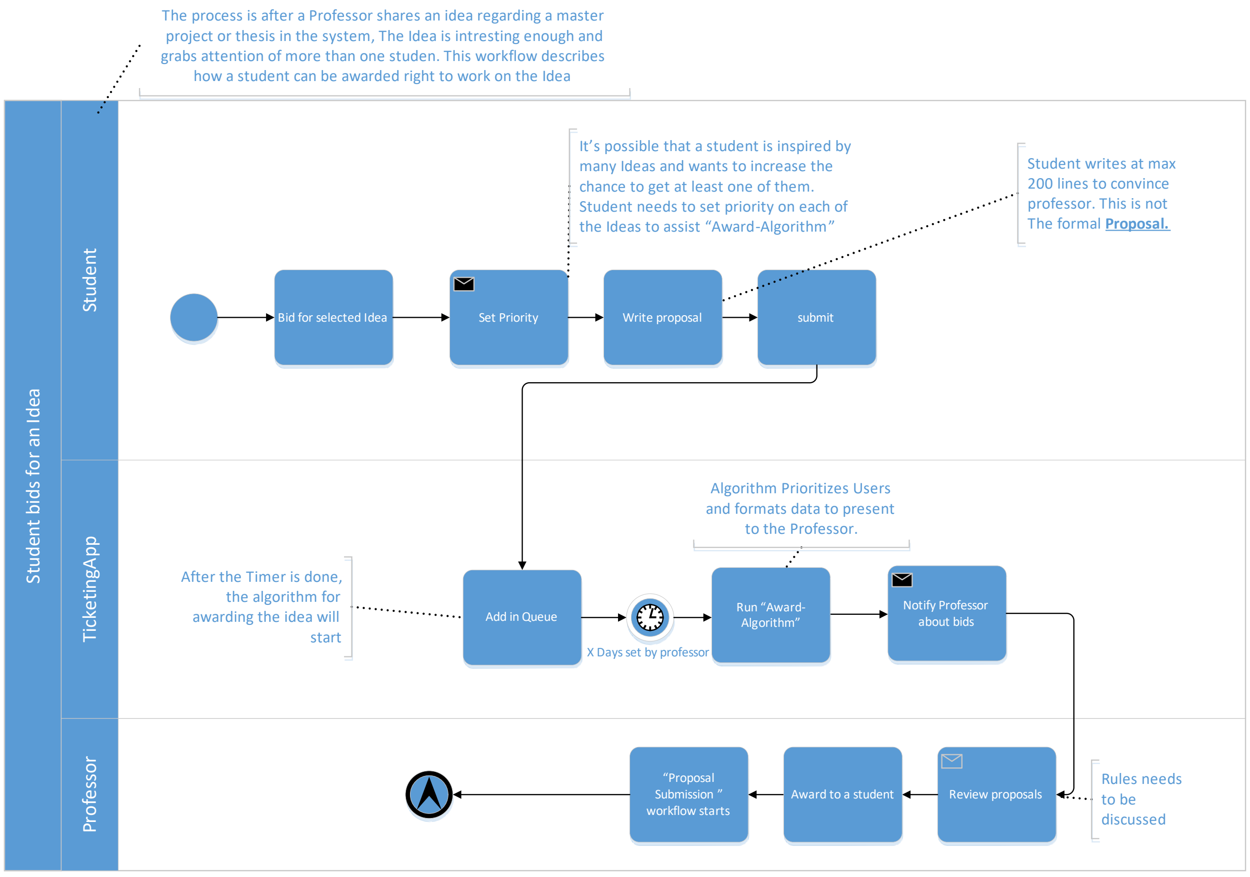
3. Professor creates an Idea

The following business process model shows how a professor creates an **idea**



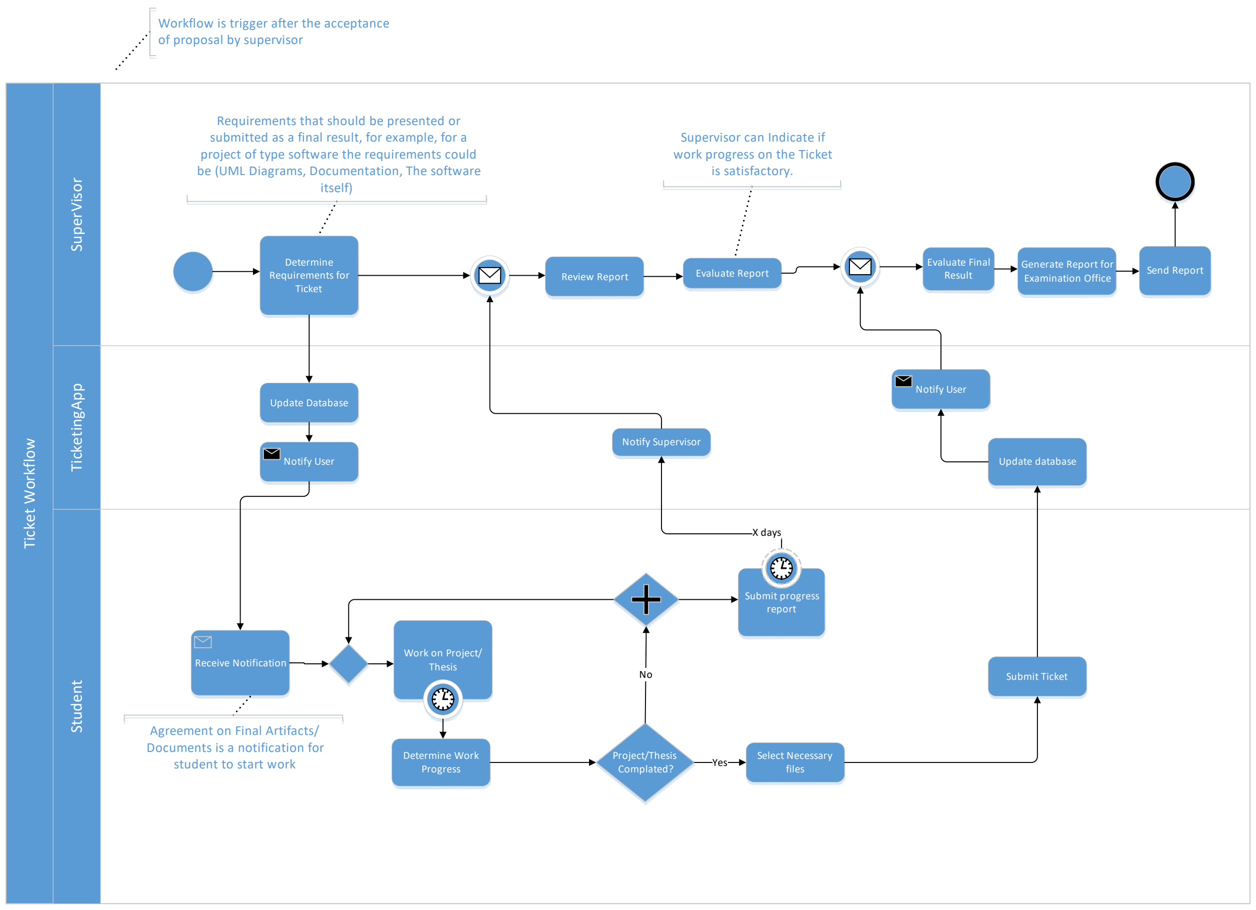
4. Bids for Available Idea:

The following business process model shows how students bid for available **ideas.**


The process is after a Professor shares an idea regarding a master project or thesis in the system, The Idea is intresting enough and grabs attention of more than one studen. This workflow describes how a student can be awarded right to work on the Idea
Student bids for an Idea
Student
TicketingApp
Professor
It's possible that a student is inspired by many Ideas and wants to increase the chance to get at least one of them. Student needs to set priority on each of the Ideas to assist "Award-Algorithm"
Student writes at max 200 lines to convince professor. This is not The formal Proposal.
Bid for selected Idea
Set Priority
Write proposal
submit
Algorithm Prioritizes Users and formats data to present to the Professor.
After the Timer is done, the algorithm for awarding the idea will start
Add in Queue
X Days set by professor
Run "Award-Algorithm"
Notify Professor about bids
"Proposal Submission " workflow starts
Award to a student
Review proposals
Rules needs to be discussed

5. Ticket Workflow

The following business process model shows how a **ticket** is produced and maintained in the system.


Workflow is trigger after the acceptance of proposal by supervisor
Ticket Workflow
SuperVisor
TicketingApp
Student
Requirements that should be presented or submitted as a final result, for example, for a project of type software the requirements could be (UML Diagrams, Documentation, The software itself)
Supervisor can Indicate if work progress on the Ticket is satisfactory.
Determine Requirements for Ticket
Review Report
Evaluate Report
Evaluate Final Result
Generate Report for Examination Office
Send Report
Update Database
Notify User
Notify Supervisor
Notify User
Update database
X days
Submit progress report
Receive Notification
Work on Project/ Thesis
Submit Ticket
No
Agreement on Final Artifacts/ Documents is a notification for student to start work
Determine Work Progress
Project/Thesis Complated?
Yes
Select Necessary files

2.5 Operating Environment

A complete overview regarding the operating environment can be obtained from the following points:

1. Programming Languages: ASP.Net MVC C#
2. UML Modeling: Microsoft Visio
3. Development Environment: Microsoft Visual Studio Community 2017
4. Database: Microsoft SQL Server (MS SQL)
5. Mockups: Balsamiq

2.6 Design and Implementation Constraints

The FH Kiel Ticketing application has the following design and implementation constraints:

- An internet connection is mandatory at all times in order to use the application. Since it is a web-based product, the user must have a working internet connection.
- Any or all **users** that will be using the system must belong to Fachhochschule Kiel
- **Users** must register their accounts on the website using their university email addresses.

2.7 User Documentation

The FH Kiel Ticketing App will have online help available for the user. The website will contain a thorough description on how to use all the important features and answer any or all questions for the user.

Apart from textual help, there will also exist a video that will illustrate how the whole process is carried out with a basic example. This form of a tutorial is selected in the project as it will provide a visual overview of the process. It will also give a better idea about the usability of the software in a shorter time.

2.8 Assumptions and Dependencies

Following factors are assumed while enlisting the features and requirements of the FH Kiel Ticketing Application:

- The first assumption is that all **users** are a part of Fachhochschule Kiel. If there are any external users then that maybe allowed through defining such **role** and a **role identifier.**
- Students that will use this system have a project or thesis to complete in their respective programs.

Users need to have a university email address in order to use the application.

3 Literature Review

3.1 Nature of Ticketing Apps

To understand the nature of ticketing applications, it is crucial that we dive into the concept of what a ticketing application, at the core level, is. This understanding would then enable us to know more about our needs and the motivation behind wanting a system that offers features like the FH Kiel Ticketing App. And in order to get into our exploration we will discuss the world of issue trackers. Why issue trackers? That is because the nature of the FH Kiel Ticketing App is similar to an issue tracker. The issue at hand are the master projects and thesis that the students need to complete. They currently have no mechanism to be tracked. Therefore the FH Kiel Ticketing App will track these issues. It will enable the users to keep track and log of what happens with every issue (in this context the projects/thesis) and who had what role in the span of an issue.

3.2 What are issue trackers

If we were to look at a text book definition of issue tracking we would find something along the lines of:

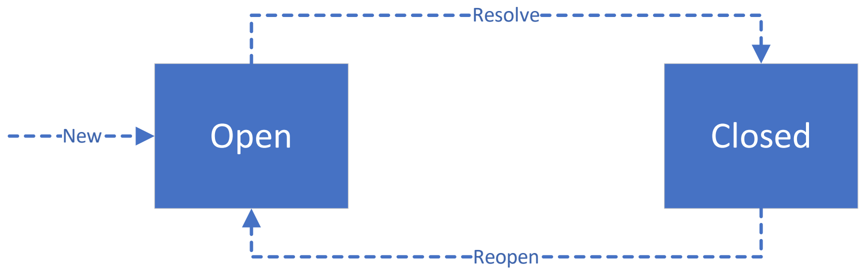
“Issue tracking, often called bug tracking (and sometimes request tracking), is the process of keeping track of your open development issues. Bug tracking is a misleading term in many ways and obviously depends on your definition of bug. “Issue” is a broad enough term to describe most of the kinds of tasks you might need to track when developing [software], and so drives our choice of terminology here.” [1]

To elaborate the text book definition, we can safely put forward the statement that it doesn't matter what terminology we use to talk about issue tracker. Whether we call it an issue tracker, bug tracker or a request tracker, at the core level they are the same class of software. This could be also said to the FH Kiel Ticketing App. The whole point of having a ticketing system is to track. Like discussed earlier, we can call the master projects and thesis as issues that need to be tracked. And these issues are not to be solved but rather tracked.[2]

3.3 Lifecycle of an Issue and Why should it be discussed

A lifecycle of an issue can be characterized by a status or a state i.e. the issue at every point of its lifecycle will belong to a certain status. These statuses might change from time to time and the issue will finally stop at a certain status. [2]

To compare it to our ticketing application, the center of attractions are tickets that act like issues which are being tracked. The tickets in the system will have certain statuses from the point it is raised to the point it is ended. It is to be noted that the end of the ticket could be a successful or an unsuccessful one. In both cases it doesn't change the fact that it is a state the ticket is in at that moment.



Simplified Lifecycle for an issue [2]

Given the above figure it can be seen that when a new issue comes up it is marked as open. This is of course a simplification of the status. Upon resolving the issue it is marked as close. However there is a possibility the issue can be reopened. The status would then be marked as open and it must be resolved again.

This is the same for the FH Kiel Ticketing application. Whenever a ticket is raised by a student, there are essentially two options for the professor. The professor either continues to work on the ticket and finish the project or thesis (depending upon the type of the ticket) or the professor can close the ticket by rejecting to work on it. This would not make the ticket useless, however this would just let it be reopened and be assigned to a new professor. Therefore the same ticket is again reopened and ready to be worked on. Therefore the lifecycle of the ticketing app is same to that of an issue tracking software.

3.4 Communicating through Issue Trackers

Apart from using issue trackers for tracking issues, it is also an excellent tool for organizations to communicate with each other through issues. According to a study done by Dane Bertram, he found out that the issue trackers used by the participants of his study was widely used for communication purposes among the members working on an issue. Each issue tracker had provided the participants a channel of communication about the issue through the form of comments. Posting comments enabled them to inform other members of an issue about the underlying topic at hand. Some issue trackers also had a “task discussion” field that would allow all stakeholders to communicate effectively with each other. [2]

Comparing it to the FH Kiel Ticketing App, the ticketing app will also provide a channel of communication inside the ticket. The ticket would not only allow professors to comment on the students’ proposals but also allow the students and professors have a general discussion which eliminates unnecessary emails.

Another thing to also keep in mind is that according to the participants of the study, they always wanted an issue tracker that was properly integrated with email services. Some users would want email notifications or reports according to customizable settings. This is the same for the FH Kiel Ticketing app. [2]

3.5 What features should Issue Trackers have

Issue trackers are supposed to provide a lot of features. Some of them should be [3]:

- The information concerned with an issue must be shared throughout the concerned members and everyone participating should know what is going on.
- Anyone going through an issue should have an instant overview of the state of the issue.
- An issue must keep history of all the actions performed within and show all changes that have happened through the existence of the issue.

All the given features are also being catered by the FH Kiel Ticketing App, therefore we can say that as an issue tracking application our ticketing application is on the right path.

3.6 What benefits Issue Trackers have

Issue trackers provide the following core benefits [3]

- It improves the quality of whatever it is intended to improve. If the issue tracker is for a software, it should help improve the quality of the software.
- It satisfies the users.
- It improves communication between the team using the software
- It increases productivity of the team

The above given benefits are also provided by the FH Kiel Ticketing App. The app improves the quality of the handling of projects/thesis between professors and students. The app consequently satisfies the users by improving communication between the professors and students and in result increasing productivity.

3.7 A glimpse at similar applications

Since there is no such application that equates exactly in functionality to that of the FH Kiel Ticketing Application, a bunch of other software were taken a look at that can be compared to the ticketing app on the basis of being an issue tracker.

3.7.1 Zendesk

Zendesk at its core level is a ticketing system that is used for tracking, prioritizing and solving customer support tickets. It places all customer support interactions in one place, which makes communication between the users seamless, personal and efficient. This results in a more productivity. Zendesk collects all information regarding tickets from their customers up-front. This makes it easy for the support team to know what kind of support a customer needs. [4]

To compare it with the FH Kiel Ticketing App, it uses a lot of information from the student up-front before raising a ticket. This information is collected in the form of a proposal document and also data that denotes the type of a ticket being raised.

3.7.2 HappyFox

HappyFox is a ticketing software that focuses on automation. They do this by having a strongly integrated email ticketing system where all incoming emails are automatically converted into tickets. Apart from emails, HappyFox also uses social media integration to convert data from, for example, Facebook to tickets in the system. [5]

The FH Kiel Ticketing App on the other hand requires no such features that HappyFox presents. HappyFox deals with customers that need constant support and therefore they need such automation that would make tickets without their manual intervention. The FH Kiel Ticketing App is for a university where the number of tickets may be high but aren't high enough to be automated into automated tickets. To add on that point, students usually have to make personalized proposals in which they present their ideas in details on which they wish to work on. Automating that will not bring any value to the product.

3.7.3 FreshDesk

FreshDesk also takes the approach of HappyFox and converts incoming emails into tickets automatically. This is because they deal with multiple customers and they wish to not miss any information coming from their end. FreshDesk also provides built-in report features where they can see statistics and identify bottlenecks. [6]

The FH Kiel Ticketing App will also have an administrator user where one can view graphs about different statistics. Just like FreshDesk states that this helps identify bottlenecks, we also believe that these graphs will let one view the bigger picture and provide really important insight at the flow of data within the organization regarding projects and thesis.

3.8 Concluding Review

Given the above reasons we can safely say that the FH Kiel Ticketing App is not only similar to an issue tracker but pretty much belongs to that class of software. The Ticketing App will provide the same features that a general issue tracker provides and along with those features it will bring forward the benefits that a general issue tracker provides.

4 External Interface Requirements

This section provides a detailed description of all inputs into and outputs from the system. It also gives a description of the hardware, software and communication interfaces and provides basic prototypes of the user interface.

4.1 User Interfaces

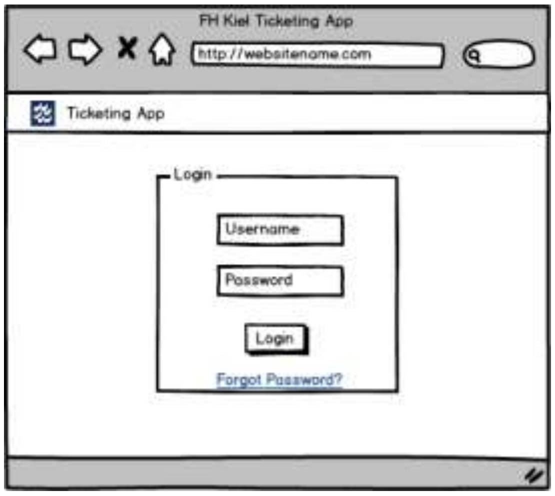
The FH Kiel Ticketing Application will be based on the standard ASP.Net MVC Application. Therefore the application will have the following properties:

- Front-end: Razer HTML
- Styling: Bootstrap v3

Since the product will be using bootstrap, the design and layout of the application shall be very straightforward and properly organized. Following mockups are a prototype of how each page might look like. It should be noted that these are mockups and are purely representing an idea of how the page shall look like. There will be elements that might be or not be a part of the page. There will be pages that might be or not be appended to the list of the provided pages.

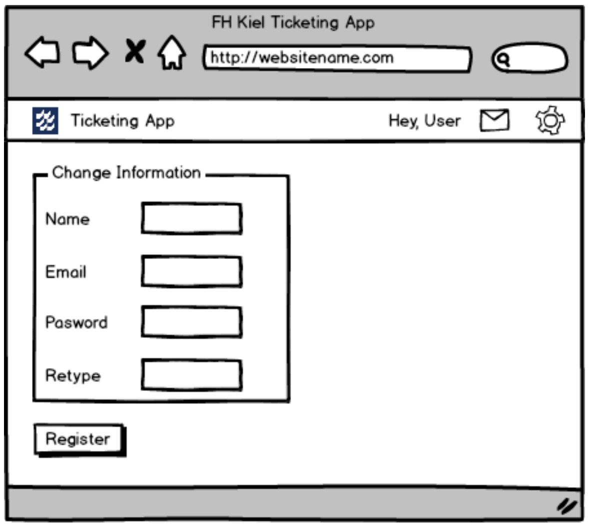
6. Login Page

This is main page and this page shall be opened up once the website is visited.



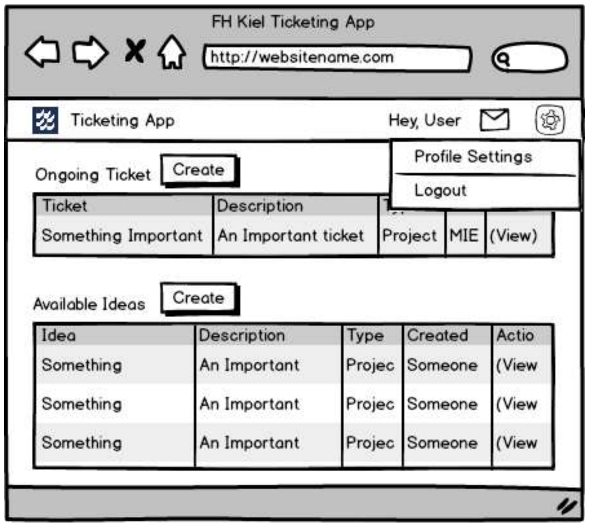
7. Register Page

This page shall be used to register the user into the system. Since the product is standalone it is required for the users (Students/Supervisors) to register their own accounts in order to use the system.



8. Home Page (after Login)

Once the user is logged in the system, this is the page that they will see.



Ongoing Ticket

Ticket	Description	T		
Something Important	An Important ticket	Project	MIE	(View)

Available Ideas

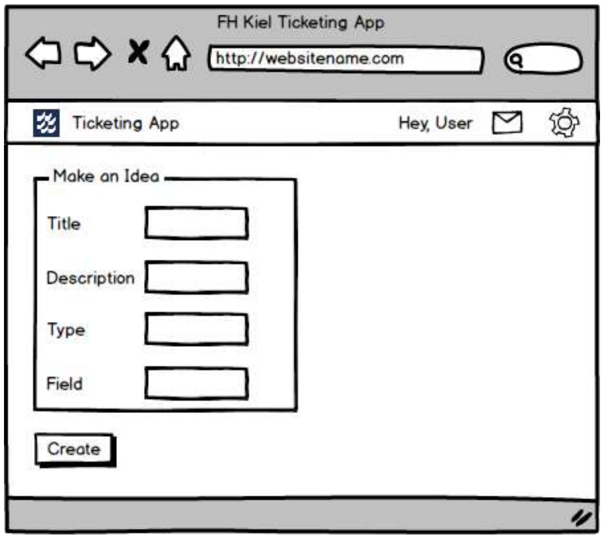
Idea	Description	Type	Created	Actio
Something	An Important	Projec	Someone	(View
Something	An Important	Projec	Someone	(View
Something	An Important	Projec	Someone	(View

9. Idea Page

In order for the **user** to make a ticket, the **user** has to make an **idea** first. The product uses this particular concept in order to make the system scalable and more general if it needs to be expanded for future uses. So, the **user** would initially make an **idea**, after wards the **idea** would become a ticket or would not become a ticket depends on the type of the **user.**

Supervisor Idea Page

When a supervisor creates an **idea** all he/she has to do is to fill in a form, specify which type it is i.e. Master Project or Master Thesis and also must specify the field it belongs to. For example an Idea could have a certain Title and Description and belongs to the field “IT-Security” and is of type “Master Project”.

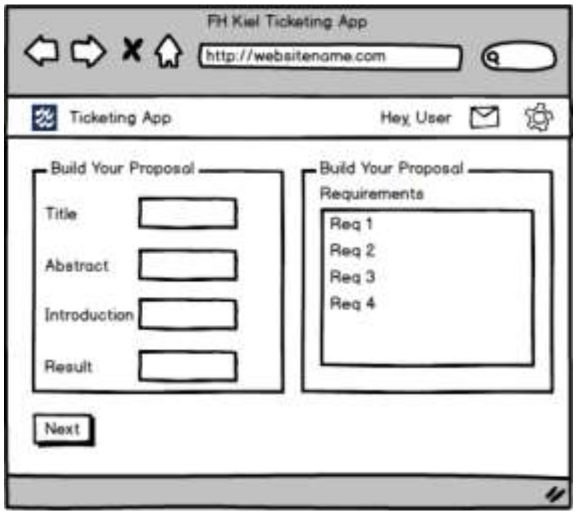


Student Idea Page

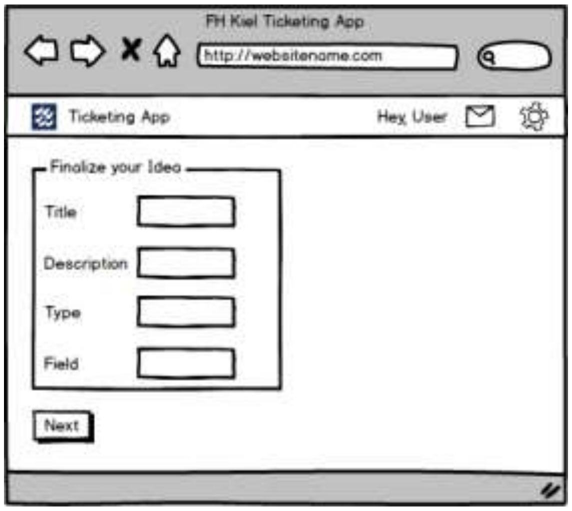
When students need to make an **idea** they have two scenarios. They are:

- Student has an **idea**
- Student needs an **idea**

In either of the two cases the student has to first of all make a Proposal. Whenever a student has to start their Project or Thesis they have to make a Proposal that would let the professor know what they are intending to do in the course of their project/thesis. The proposal follows a standard template for all the Professors. They expect a similar type of document and to ensure that, the students will make their proposal on the website to which they will hold the ability to export to PDF.



Once they are done with their proposal they will then finalize their **ideas.**

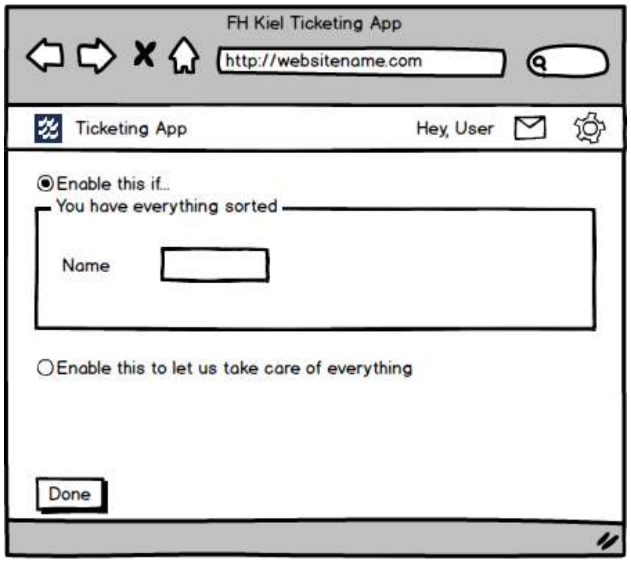


10. Idea – to – Ticket Transition

Once the students are done finalizing their **ideas**¸ then they are in the middle point to where their **idea** will then become a **ticket**.

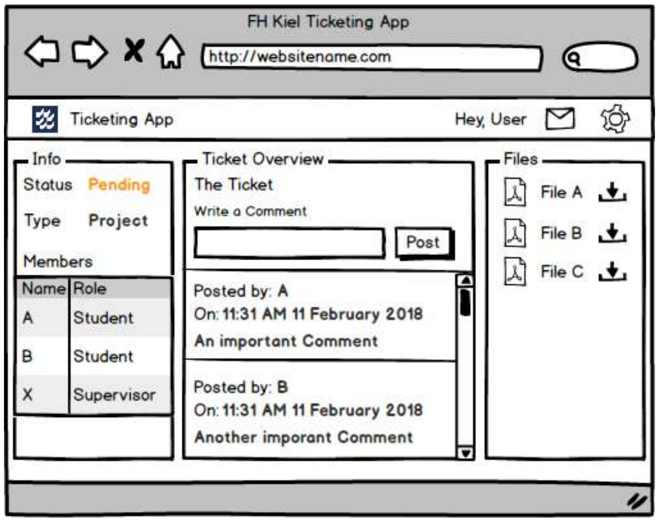
This middle point helps the student transition from an **idea** to a **ticket**. In this transition state the students have to specify some more things in order to help the system choose the correct supervisor for them. Therefore the following two possibilities are covered:

1. The student has already discussed something with a professor and wants to work with someone specific. In this case they will write the name of the professor. They have a free hand here and it can be argued that this may be misused in a way that the students might just write down a professor's name with whom they have discussed nothing. The system takes care of this later. The **ticket** has a counter which shows how many times it has been rejected. If the student were to misuse this feature they will get a rejection marked on their **ticket** which is going to be bad for them.
2. In another scenario the student has not discussed with any professor, in which case they check another radio button through which the system will redirect the **ticket** to a professor that matches the criteria of the project.



11. Ticket Page

Once the **ticket** is created, it is then showed in a dashboard type of a view. In this page, all members of the **ticket** are shown. Along with that, it also shows all the comments and files uploaded in the **ticket**, only the members of the **ticket** are authorized to view the **ticket**



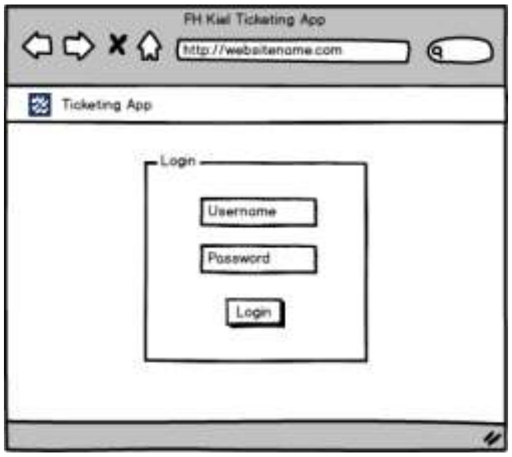
Name	Role
A	Student
B	Student
X	Supervisor

Posted by: A
On: 11:31 AM 11 February 2018
An important Comment

Posted by: B
On: 11:31 AM 11 February 2018
Another imporant Comment

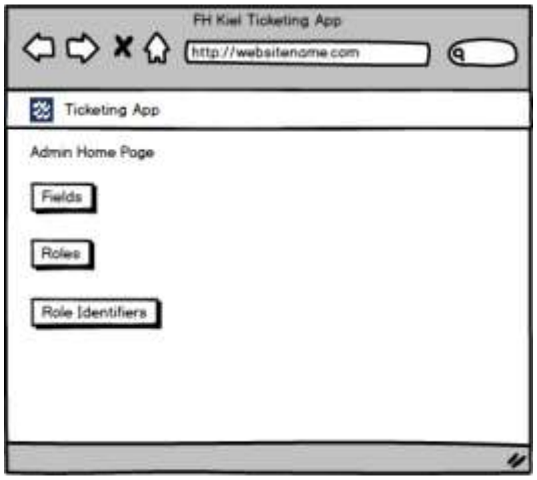
12. Administrator Login

The **administrator** can log in using this page. Since the **administrator** has a different purpose in the system, therefore he/she gets a different login page.



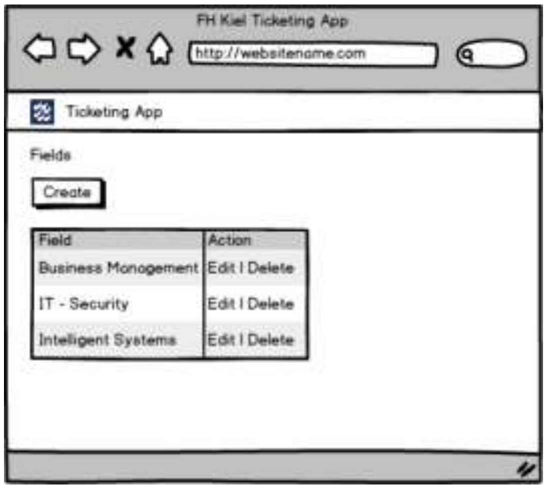
13. Admin Home Page

Once the **administrator** logs in, he/she will see the following page where they can perform CRUD (create, read, update, delete) operations on **Fields, Roles,** and **Role Identifiers.**

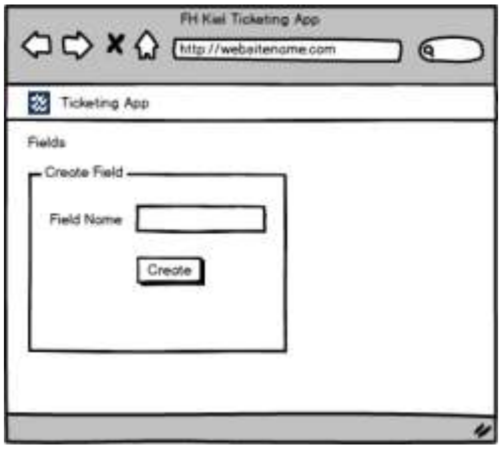


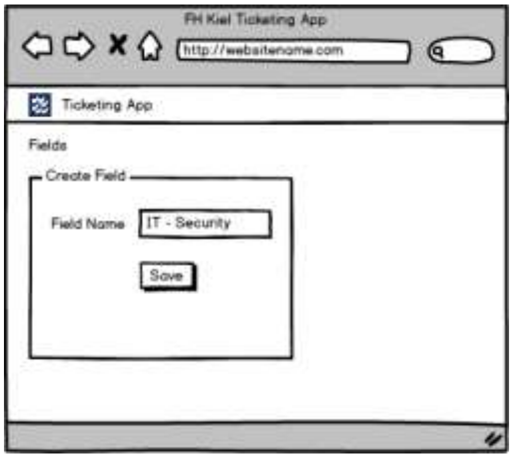
14. Field – CRUD

The following screen shows how the system handles the CRUD operations for **Fields.**



Field	Action
Business Management	Edit I Delete
IT - Security	Edit I Delete
Intelligent Systems	Edit I Delete

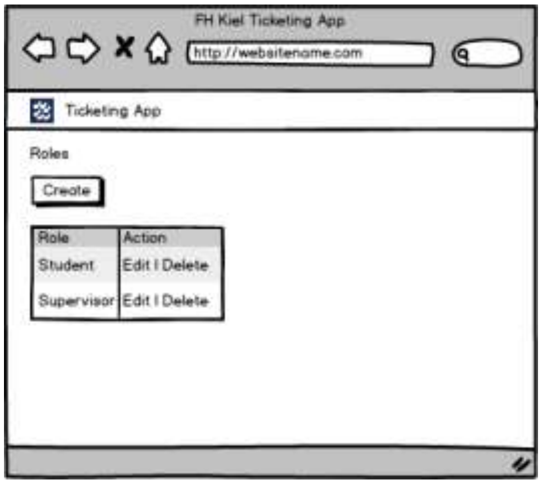




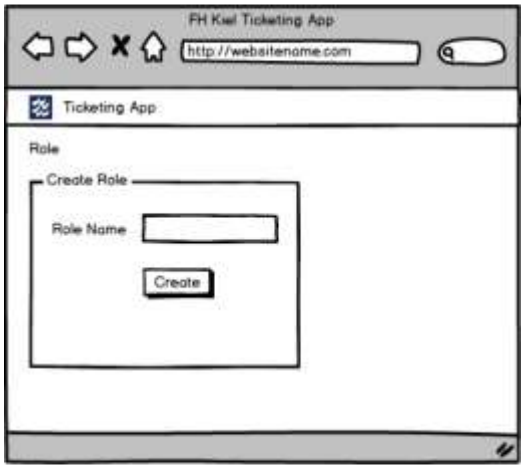
Field Name: IT - Security

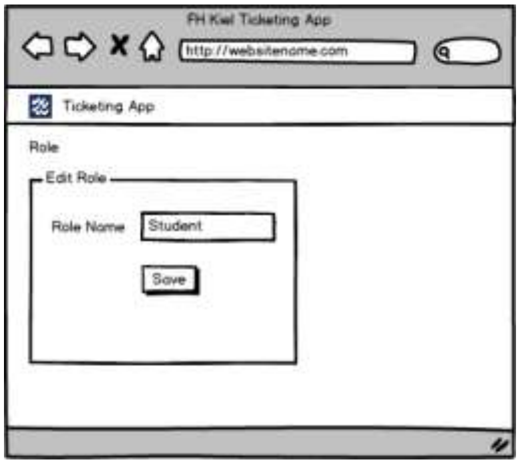
15. Roles – CRUD

The following screen shows how the system handles the CRUD operations for **Roles.**



Role	Action
Student	Edit I Delete
Supervisor	Edit I Delete

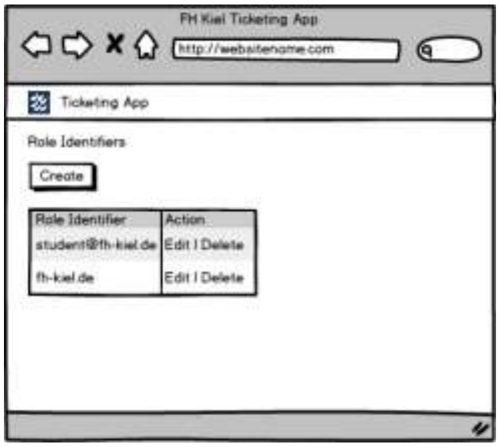




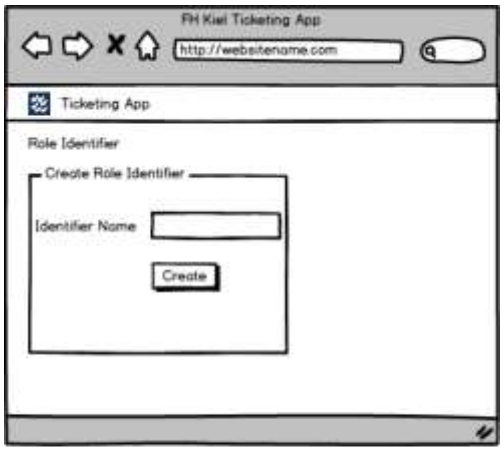
Role Name: Student

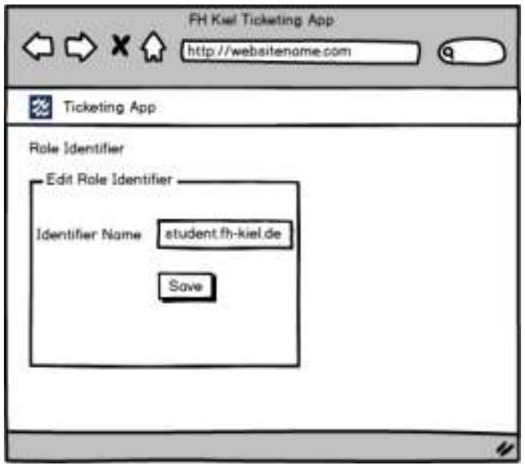
16. Role Identifier – CRUD

The following screen shows how the system handles the CRUD operations for **Role Identifier.**



Role Identifier	Action
student@fh-kiel.de	Edit I Delete
fh-kiel.de	Edit I Delete





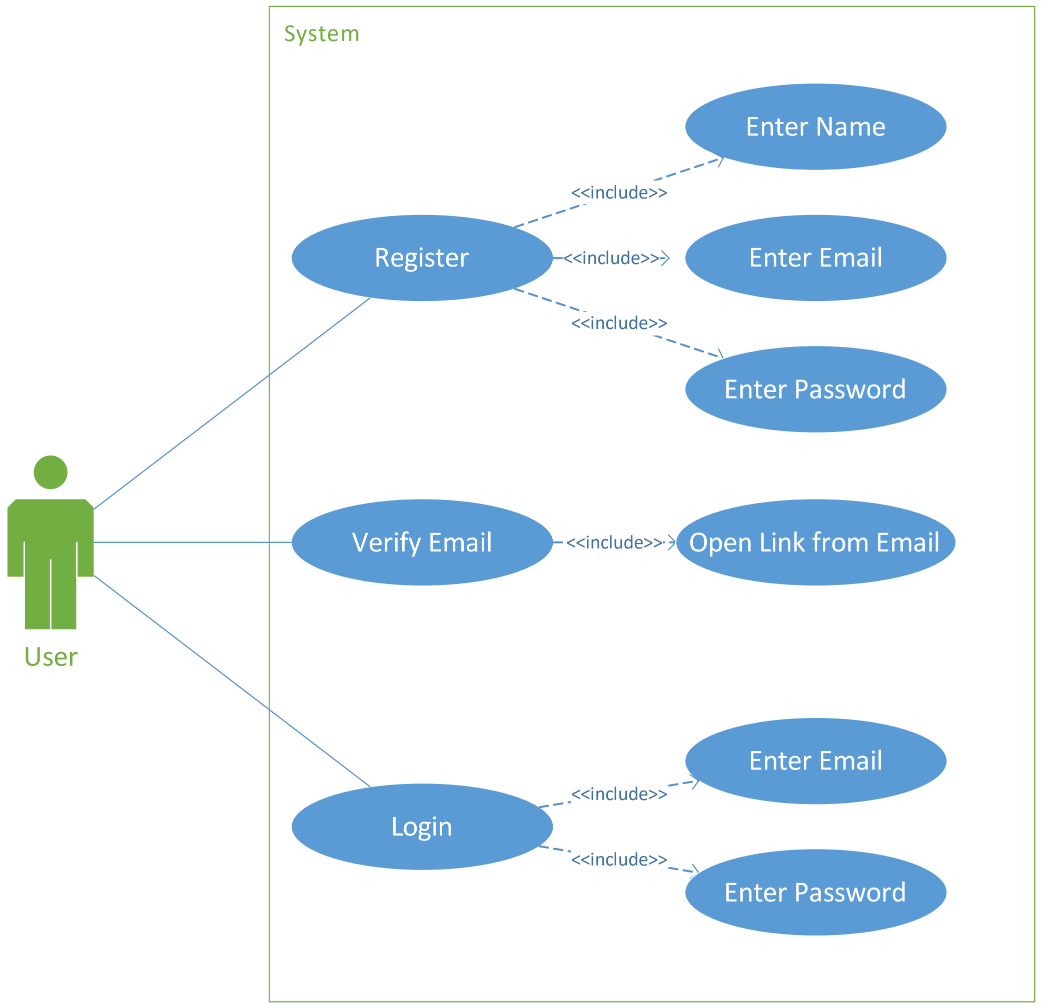
Identifier Name: student.fh-kiel.de

17. User Interfaces – Use Cases

The following segment of the document contains any and all use cases for the system. Along with the use cases you will find a use case description for more details.

Login/Register

The following use case demonstrates the activities that happen when a **user** either registers or logs in.



Use Case Name:	*Register*
Primary Actor	**User**
Trigger	**User** will trigger this use case
Precondition	• **User** has internet connection. • **User** is connected to our system.

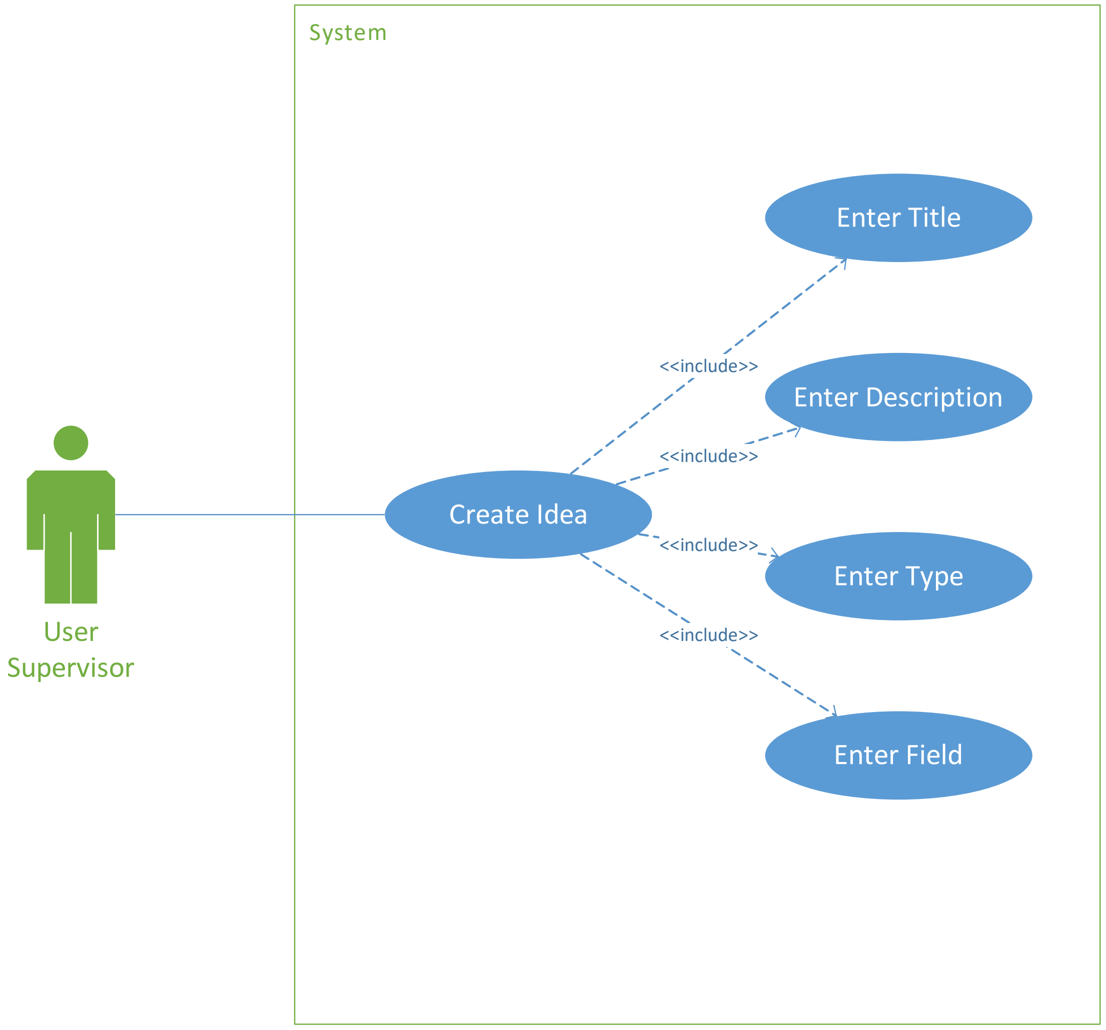
Main Success Scenario	1. **User** enters name 2. **User** enters email 3. **User** enters password 4. **User** retypes same password
Alternate Path	N/A
Post-condition	**User** gets a message indicating to check email for verification link
Extensions	N/A

Use Case Name:	*Verify Email*
Primary Actor	**User**
Trigger	**User** will trigger this use case
Precondition	• **User** has internet connection. • **User** is connected to our system.
Main Success Scenario	1. **User** clicks on activation link from email
Alternate Path	N/A
Post-condition	**User** gets a message indicating to login since now the **user's** account is activated
Extensions	N/A

Use Case Name:	*Login*
Primary Actor	**User**
Trigger	**User** will trigger this use case
Precondition	• **User** has internet connection. • **User** is connected to our system.
Main Success Scenario	1. **User** enters email 2. **User** enters password
Alternate Path	N/A
Post-condition	**User** logs in and sees their respective home page.
Extensions	N/A

Supervisor Creates Idea

The following use case discusses the scenario where a supervisor creates an **idea** that can be later utilized by a student.

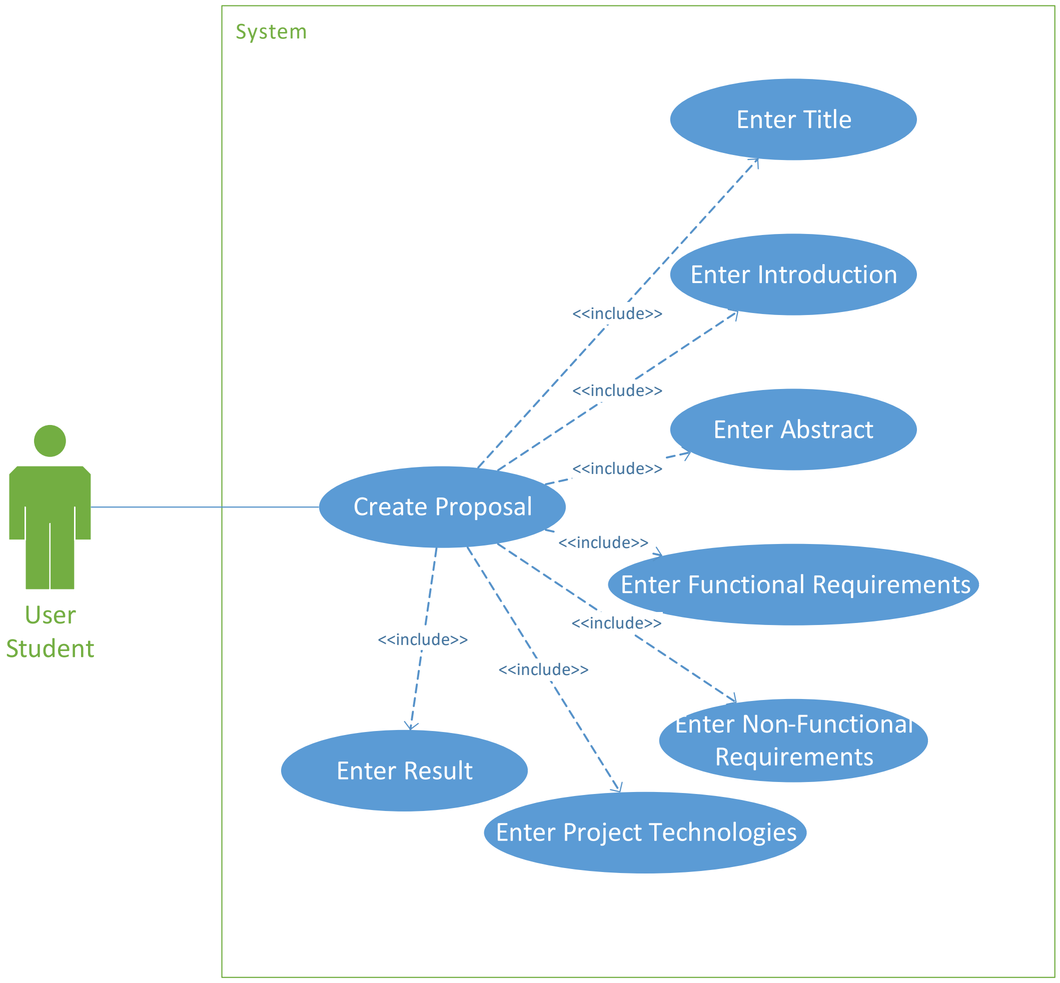


Use Case Name:	*Create Idea*
Primary Actor	**User – Supervisor**
Trigger	**User – Supervisor** will trigger this use case
Precondition	• **User** has internet connection. • **User** is connected to our system.
Main Success Scenario	1. **User** enters title of the **idea** 2. **User** enters description of the **idea** 3. **User** enters type of the **idea** (Master Project/Thesis) 4. **User** enters field of the **idea**
Alternate Path	N/A

Post-condition	**User** views the **idea** that has been created.
Extensions	N/A

Student Creates Idea

This use case discusses how a student initiates to create an **idea** and what are the steps involved.

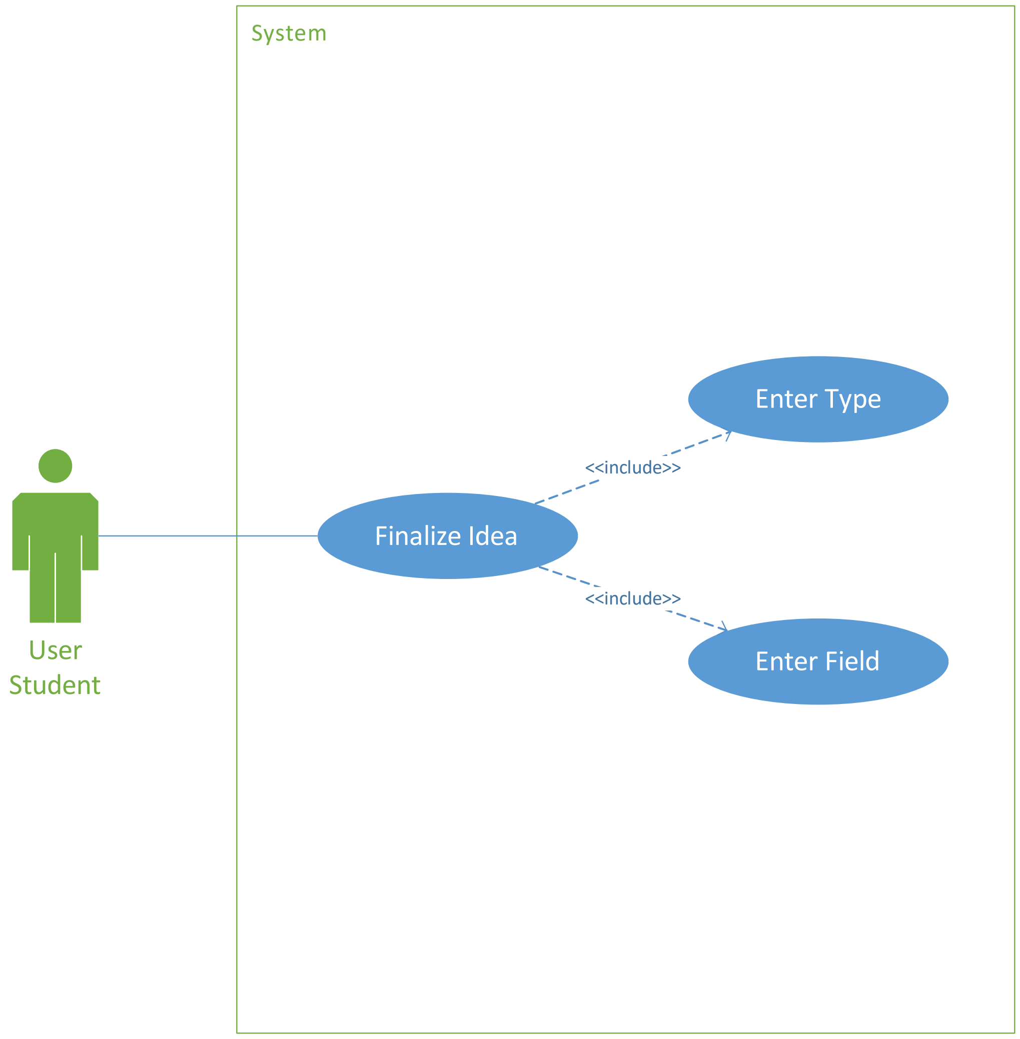


Use Case Name:	*Create Proposal*
Primary Actor	**User – Student**
Trigger	**User – Student** will trigger this use case
Precondition	• **User** has internet connection. • **User** is connected to our system.
Main Success Scenario	1. **User** enters title of the Proposal 2. **User** enters abstract of the Proposal

	3. **User** enters introduction of the Proposal 4. **User** enters functional requirements of the Proposal 5. **User** enters non-functional requirements of the Proposal 6. **User** enters project technologies 7. **User** enters result
Alternate Path	N/A
Post-condition	**User** moves forward to finalize the **idea**
Extensions	N/A

Student Finalizes Idea

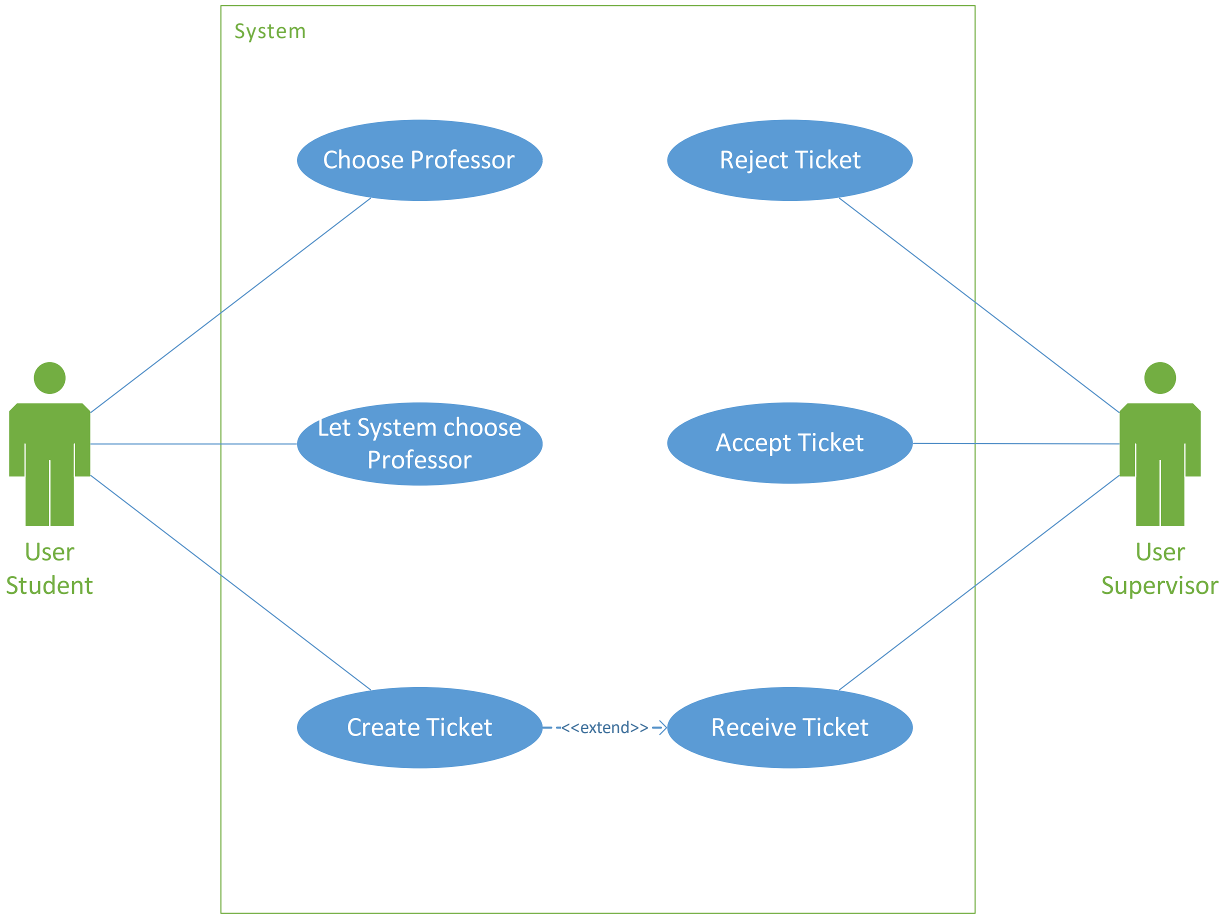
The following use case discusses the scenario where a student has created a proposal and is now going to finalize an **idea** which essentially means creating an **idea.** The **user** does not need to enter any title or description since he/she has already made a proposal which contains the relevant data. Instead the **user** only has to enter the type of the **idea** and which **field** it belongs to.



Use Case Name:	*Finalize **Idea***
Primary Actor	**User – Student**
Trigger	**User – Student** will trigger this use case
Precondition	• **User** has internet connection. • **User** is connected to our system.
Main Success Scenario	1. **User** enters type of the **idea** (Master Project/Thesis) 2. **User** enters field of the **idea**
Alternate Path	N/A
Post-condition	**User** views the **idea** that has been created.
Extensions	N/A

Idea to Ticket Transition

The following use case shows the transition the system makes to transform the **idea** to a **ticket.**



Use Case Name:	*Choose Professor*
Primary Actor	**User – Student**
Trigger	**User – Student** will trigger this use case
Precondition	• **User** has internet connection. • **User** is connected to our system. • **User – Student** already has discussed with a professor
Main Success Scenario	1. **User** enters name of Professor
Alternate Path	N/A
Post-condition	**User** views the **ticket** has been created
Extensions	N/A

Use Case Name:	*Let System choose Professor*
Primary Actor	**User – Student**
Trigger	**User – Student** will trigger this use case
Precondition	• **User** has internet connection. • **User** is connected to our system. • **User – Student** has not discussed with a professor.
Main Success Scenario	1. **User** chooses a radio button on screen
Alternate Path	N/A
Post-condition	**User** views the **ticket** has been created
Extensions	N/A

Use Case Name:	*Create Ticket*
Primary Actor	**User – Student**
Trigger	**User – Student** will trigger this use case
Precondition	• **User** has internet connection. • **User** is connected to our system. • **User – Student** has either chosen a professor or has let the system choose one
Main Success Scenario	1. System creates a **ticket**
Alternate Path	N/A
Post-condition	**User** views the **ticket** screen
Extensions	N/A

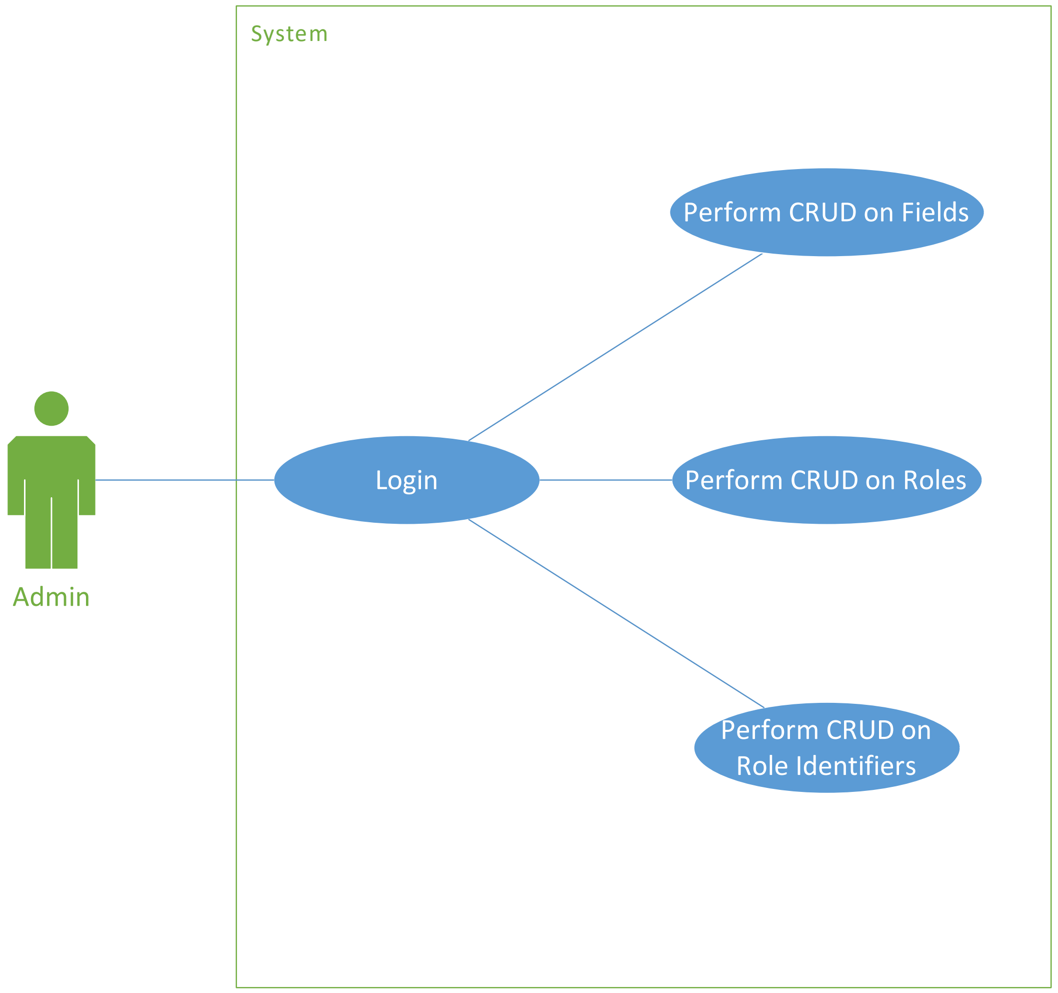
Use Case Name:	*Receive Ticket*
Primary Actor	**User – Supervisor**
Trigger	**User – Supervisor** will trigger this use case
Precondition	• **User** has internet connection. • **User** is connected to our system. • **User – Student** has created a **ticket**
Main Success Scenario	1. **User** receives a notification of a **ticket**
Alternate Path	N/A
Post-condition	**User** views the **ticket** screen
Extensions	N/A

Use Case Name:	*Accept Ticket*
Primary Actor	**User – Supervisor**
Trigger	**User – Supervisor** will trigger this use case
Precondition	• **User** has internet connection. • **User** is connected to our system. • **User – Supervisor** has received a **ticket**
Main Success Scenario	1. **User** clicks on accept
Alternate Path	N/A
Post-condition	**User** views the **ticket** screen and status has changed
Extensions	N/A

Use Case Name:	*Reject Ticket*
Primary Actor	**User – Supervisor**
Trigger	**User – Supervisor** will trigger this use case
Precondition	• **User** has internet connection. • **User** is connected to our system. • **User – Supervisor** has received a **ticket**
Main Success Scenario	2. **User** clicks on reject
Alternate Path	N/A
Post-condition	**User** views the **ticket** screen and status has changed
Extensions	N/A

Admin Panel

The following use case shows the functionality of the admin panel performed by the **administrator.**



Use Case Name:	*Login*
Primary Actor	**Administrator**
Trigger	**Administrator** will trigger this use case
Precondition	• **Administrator** has internet connection. • **Administrator** is connected to our system.
Main Success Scenario	1. **Administrator** provides correct username and password
Alternate Path	N/A
Post-condition	**Administrator** views the Admin Home Page
Extensions	N/A

Use Case Name:	*Perform CRUD on **Fields***
Primary Actor	**Administrator**

Trigger	**Administrator** will trigger this use case
Precondition	• **Administrator** has internet connection. • **Administrator** is connected to our system. • **Administrator** is logged in
Main Success Scenario	1. **Administrator** views the list of **Fields** 2. **Administrator** can create a new **Field** 3. **Administrator** can edit a **Field** 4. **Administrator** can delete a **Field**
Alternate Path	N/A
Post-condition	**Administrator** views the list of **Fields**
Extensions	N/A

Use Case Name:	*Perform CRUD on **Roles***
Primary Actor	**Administrator**
Trigger	**Administrator** will trigger this use case
Precondition	• **Administrator** has internet connection. • **Administrator** is connected to our system. • **Administrator** is logged in
Main Success Scenario	5. **Administrator** views the list of **Roles** 6. **Administrator** can create a new **Role** 7. **Administrator** can edit a **Role** 8. **Administrator** can delete a **Role**
Alternate Path	N/A
Post-condition	**Administrator** views the list of **Roles**
Extensions	N/A

Use Case Name:	*Perform CRUD on **Role Identifiers***
Primary Actor	**Administrator**
Trigger	**Administrator** will trigger this use case
Precondition	• **Administrator** has internet connection. • **Administrator** is connected to our system. • **Administrator** is logged in
Main Success Scenario	9. **Administrator** views the list of **Role Identifiers** 10. **Administrator** can create new **Role Identifiers** 11. **Administrator** can edit **Role Identifiers** 12. **Administrator** can delete **Role Identifiers**
Alternate Path	N/A

Post-condition	**Administrator** views the list of **Role Identifiers**
Extensions	N/A

4.2 Hardware Interfaces

Since the product is a web based application therefore it has no such hardware interface. Although if it were mandatory to name some hardware interfaces one can say that a computer with an internet connection and a browser will suffice to use the application.

4.3 Software Interfaces

The product communicates mainly with a database. The database includes data of the **users** and as well as their concerned items. The product will not only read data from the database but also write data. Other than the database the software neither sends nor receives data to any third party library or interface that may or may not modify the behavior of the system.

4.4 Communications Interfaces

The FH Kiel Ticketing App requires a stable internet connection and a web browser to be properly used. The communication is mainly handled by the browser's HTTP. The product has no FTP communication.

5 System Features

Following are all the system features that the FH Kiel Ticketing Application will hold. The purpose of these features being listed here is to provide a clear understanding of any and all properties that the system will contain and to showcase any and all capabilities that the system should be capable of to be called complete.

5.1 Administration Panel

The administration panel allows a specific user to change the system configuration.

18. Description and Priority

The FH Kiel Ticketing Application is a big application and has many elements in the database that have to be predefined in order for the system to function. For this purpose the administration panel comes into play where it provides a specific user to manipulate these settings and perform basic CRUD (create/read/update/delete) operations.

19. Stimulus/Response Sequences

The **administrator** uses this feature in the system. See 3.1.12.6. Use Case: Perform CRUD on **Fields** & Perform CRUD on **Roles** & Perform CRUD on **Role Identifiers**

20. Functional Requirements

Following are the functional requirements for the stated system feature.

REQ - 1 The system shall allow the **administrator** to login to the administration panel.

REQ - 2 The system shall let the **administrator** perform CRUD (create, read, update, delete) operations on the following:

a. The **administrator** shall be able to perform CRUD on the **Fields** that will be used in the system.
b. The **administrator** shall be able to perform CRUD on the **Roles** that will be used in the system.
c. The **administrator** shall be able to perform CRUD on the **Role Identifiers** that will be used in the system.

REQ - 3 The system shall let the **administrator** see the following statistics as graphs:

a. The system shall show the **administrator** the number of proposals in the system.
b. The system shall show the **administrator** the number of tickets in the system.
    i. The system shall show the **administrator** the number of accepted **tickets.**
    ii. The system shall show the **administrator** the number of rejected **tickets.**
    iii. The system shall show the **administrator** the number of **tickets** as master projects.
    iv. The system shall show the **administrator** the number of **tickets** as thesis.

5.2 Register & Login

The register and login feature allows the **users** to become a part of the system and to use it.

21. Description and Priority

Since the FH Kiel Ticketing App is a standalone system, the **users** that are to be a part of the system must register themselves in order to use it. After registration they must login with their registered accounts.

22. Stimulus/Response Sequences

The **user** triggers this action in the system. They fill in a form and must verify their email after submitting the form. See 3.1.12.1. Use Case: Register & Use Case: Login

23. Functional Requirements

Following are the functional requirements for the stated system feature.

REQ - 1 The system shall allow the **users** to register their account.
REQ - 2 The system shall allow the **users** to enter their *data* [c]
REQ - 3 The system shall let the **users** verify their accounts.
REQ - 4 The system shall identify the **user** type (student or supervisor) based on their email address.
REQ - 5 The system shall let the **users** log in to the system using their registered accounts.
REQ - 6 The system shall allow the **users** to stay logged in for a 30 minutes.
REQ - 7 The system shall allow the **users** to reset their passwords.

5.3 Creating Idea - Supervisor

24. Description and Priority

The Supervisor can create **ideas** on the system that can be used by the students for their master projects or thesis.

25. Stimulus/Response Sequences

The **user - supervisor** triggers this action in the system. They fill in a form and must submit it. See 3.1.12.2. Use Case: Create Idea

26. Functional Requirements

Following are the functional requirements for the stated system feature.

REQ - 1 The system shall allow the **user-supervisor** to perform CRUD (create/read/update/delete) operations on the following:

a. The **user-supervisor** shall be able to perform CRUD operations on **Ideas.**

5.4 Creating/Picking Idea - Student

27. Description and Priority

The student can either create an **idea** or choose one that a Professor has made.

28. Stimulus/Response Sequences

The **user - student** triggers this action in the system. They fill in a form and must submit it. See 3.1.12.3. Use Case: Create Proposal

29. Functional Requirements

The following are the functional requirements for the given feature

REQ - 1 The system shall allow the **user – student** to create an **Idea.**
REQ - 2 The system shall let the **user – student** create a proposal for their **idea.**
REQ - 3 The system shall let the **user – student** choose whether the **idea** is for a thesis or a master project.
REQ - 4 The system shall let the **user – student** choose the **field** of the **idea.**
REQ - 5 The system shall let the **user – student** choose a Supervisor for their **idea.**
REQ - 6 The system shall allow the **user – student** to let the system choose **supervisor.**
REQ - 7 The system shall let the **user – student** view their created **idea** and **proposal.**

5.5 Idea to Ticket - Transition

30. Description and Priority

Once the student is done with the **idea** and the **proposal.** The system automatically raises a **ticket** out of the created **idea and proposal.**

31. Stimulus/Response Sequences

The **user - student** triggers this action in the system. They create an **idea** and a **proposal** which becomes a **ticket**. See 3.1.12.5. Use Case: Create Ticket

32. Functional Requirements

The following are the functional requirements for the given feature

REQ - 1 The system shall automatically create a **ticket** from the **idea** and **proposal** created by the student.
REQ - 2 The system shall automatically include the **proposal** in the **ticket.**
REQ - 3 The system shall inform the chosen Supervisor about the created **ticket.**
REQ - 4 The system shall provide the Professor the ability to reject the **ticket.**
REQ - 5 The system shall provide the Professor the ability to accept the **ticket.**
REQ - 6 The system shall inform the examination department with *adequate data* [e] about the accepted **ticket** if the **ticket** were to be a Thesis. (This requirement is dependent on REQ – 5).

5.6 Ticket

33. Description and Priority

Once a **ticket** is raised, the members of the **ticket** can now perform many functions through this. They will be provided with a ticket-area where they can upload files and communicate with one another.

34. Stimulus/Response Sequences

The **user - student** has already triggered this action in the system. They create an **idea** and a **proposal** which becomes a **ticket**. See 3.1.12.5. Use Case: Create Ticket

35. Functional Requirements

The following are the functional requirements for the given feature

REQ - 1 The system shall show all the **Users** that belong to the **ticket.**
REQ - 2 The system shall show the current **status** of the ticket.
REQ - 3 The system shall show all the files that are uploaded for the **ticket.**
REQ - 4 The system shall show all the **comments** inside the **ticket.**
REQ - 5 The system shall show the current progress of the **ticket.**
REQ - 6 The system shall show the *final files* [d] that need to be submitted in the **ticket.**
REQ - 7 The system shall allow the members of the **ticket** to upload new files.
REQ - 8 The system shall allow the members of the **ticket** to post new **comments.**
REQ - 9 The system shall allow the members of the **ticket** to view the current progress of the **ticket.**
REQ - 10 The system shall allow the Supervisor to change the **status** of the **ticket.**

a. The supervisor can request for more changes in the **proposal** before accepting to supervise the **ticket.**
b. The supervisor can close the **ticket** for the following two possible reasons:
    i. The **idea** is not good enough to be pursued as a project/thesis
    ii. The project/thesis is now complete and *final files* [4] have been submitted.

REQ - 11 The system shall allow the members of the **ticket** to view the created **proposal.**
REQ - 12 The system shall allow the Supervisor to **comment** on the created **proposal.**
REQ - 13 The system shall allow the Student to edit the **proposal.**
REQ - 14 The system shall maintain a history of past **proposals** in the **ticket.**
REQ - 15 The system shall allow other Students request to join the **ticket**.
REQ - 16 The system shall allow the Supervisor to add another Supervisor in the **ticket.**
REQ - 17 The system shall allow the Supervisor to let other students join the **ticket.**

5.7 Ticket Rejection

36. Description and Priority

There can be a scenario where Professor might not want to work on the **ticket**. In that case the Professor/Supervisor will reject the **ticket.**

37. Stimulus/Response Sequences

The **user - supervisor** will trigger this action in the system. They reject a **ticket** by giving a reason. See 3.1.12.5. Use Case: Reject Ticket

38. Functional Requirements

The following are the functional requirements for the given feature

REQ - 1 The system shall allow the Supervisor to provide a reason for rejecting a **ticket.**
a. The system shall allow the Supervisor to reject the **ticket** if they don't have time to pursue it.
b. The system shall allow the Supervisor to reject the **ticket** if they think the **idea** is not eligible to be a project or thesis.

REQ - 2 The system shall keep count of how many times a **ticket** has been rejected.
REQ - 3 The system shall close the **ticket** if it has been rejected three times because the **idea** was not eligible to be a project or thesis.
REQ - 4 The system shall inform the Student to make a new **ticket** if their previous **ticket** is rejected three times because the **idea** was not eligible to be a project or a thesis.
REQ - 5 The system shall allow a Supervisor to assign a **ticket** to him/herself if the **ticket** has been rejected three times due to Supervisors not having time.

5.8 Home Screen

39. Description and Priority

In a huge system such as this it is imperative that the **users,** when logged in, can see an overall view of what is going on around them.

40. Stimulus/Response Sequences

Both **user** types will have this feature. There is no particular use case for this since this is just a view.

41. Functional Requirements

The following are the functional requirements for the given feature

REQ - 1 The system shall show the **user – supervisor** all **tickets** he/she is a part of.
 a. The system shall show the **user – supervisor** all **tickets** that he/she is supervising.
 b. The system shall show the **user – supervisor** all **tickets** that he/she needs to check-out whether he/she wants to supervise or not.

REQ - 2 The system shall show the **user – supervisor** all the **ideas** he/she has made

REQ - 3 The system shall show the **user – student** the current **ticket** he/she is a part of.

REQ - 4 The system shall show the **user – student** all the **ideas** in the system.
 a. The system shall show the **user – student** all the **ideas** he/she can join.
 b. The system shall show the **user – student** all the **ideas** he/she cannot join.

REQ - 5 The system shall allow the **user** – **student** to filter **ideas.**

REQ - 6 The system shall show the **user – student** his/her created **proposals.**

5.9 Notifications

42. Description and Priority

The system will send out notifications that the **users** can view within the website. These notifications will also be sent per mail.

43. Stimulus/Response Sequences

Both **user** types will have this feature. There is no particular use case for this since most of the actions in the system will trigger a notification to the concerned **users.**

44. Functional Requirements

The following are the functional requirements for the given feature

REQ - 1 The system shall provide the **user** to view notifications.
 a. The system shall show notifications for updates in concerned **tickets.**
  i. The system shall show notifications when **user** uploads a file in a **ticket.**
  ii. The system shall show notifications when **user** posts **comments** in a **ticket.**
  iii. The system shall show notifications when **ticket's status** has been changed.
 b. The system shall show notifications for requests in concerned **tickets.**
  i. The system shall show notifications if someone wants to join a **ticket.**
  ii. The system shall show notifications to concerned **users** if **user – student** has raised a **ticket.**

REQ - 2 The system shall provide the **user** the ability to receive email notifications (See REQ – 1 for the detail of notification types)

REQ - 3 The system shall provide the **user – supervisor** the ability to receive automatic report emails after a certain number of days.

a. The system shall allow the **user – supervisor** to choose the number of days after which he/she would like a report email.
b. The system shall show the number of new **tickets** raised for the **user – supervisor** in the report email
c. The system shall show the number of **tickets** updated that the **user – supervisor** is a part of.

6 User Stories

To understand the system more clearly, following are some user stories that inform you about the functionalities of the system, provide more clarification and puts forward a broader picture.

User Story

Content	As a **user** I would like to system to recognize me automatically as a student or a professor based on my data.
Acceptance Criteria	• The system sends an email to verify the user • The system informs the **user** about the his/her user type

User Story

Content	As a **user - supervisor** I want to see a list of **proposals** which are under my supervision so that it is easy to keep track of each one of them.
Acceptance Criteria	• The list should be ordered in a way so that the new **proposals** are on top followed by the ongoing ones and then finally the completed projects/thesis. • Each **proposal** when clicked should redirect to page with full description about the project/thesis.

User Story

Content	As a **user - student** I should be able to create my **proposal** for master project/thesis using the system itself so that it can be submitted through the system and doesn't require uploading a PDF as a **proposal**.
Acceptance Criteria	• There should be an option to select the type of **proposal** i.e. Project or Thesis. • All the required information for a **proposal** can be filled including type of field, abstract, introduction, requirements, technologies required, conclusion and references. • There should be possibility to select the professor under whose supervision the student wants to do the project/thesis o There should be a way where the professor can be chosen by the student if the project has already been discussed between them or, o The system should generate a list of all the available professors from where the student can choose. • The submitted **proposal** should be displayed on the chosen professor's dashboard so that he can take further action

User Story

Content	As a **user - supervisor** I should be able to view the **proposal** details when I click on any **proposal** so that it can be reviewed.

Acceptance Criteria	• All the information in the **proposal** can be viewed and feedback (as **comments)** can be given.
	• The **status** of the **ticket** can be changed by the professor based on the following list:
	  o Open
	  o Proposal awaiting approval
	  o Amendment required
	  o Proposal rejected
	  o Proposal accepted
	  o Ticket in progress
	• Ability to see progress of the project/thesis.
	• Ability to view the history of all the actions taken place when the **proposal** is active

User Story

Content	As a **user - student** I should be able to view the **ticket** details after raising it so that I can see the actions the professor takes.
Acceptance Criteria	• Ability to see progress of the project/thesis. • Ability to view the history of all the actions taking place. • Ability to upload documents to the system. • Ability to interact with the professor through the system

User Story

Content	As a **user - supervisor** I should be able to post an **idea** so that the interested students can submit their proposals.
Acceptance Criteria	• Ability to fill the information like the name of the project, type, field and description.

User Story

Content	As a **user - student** I should be able to upload biweekly reports to the system so that the professor can review it.
Acceptance Criteria	• Ability to upload documents. • Ability to view the professor's **comments** • Ability to write **comments**

User Story

Content	As a **user - supervisor** I should be able to view the reports submitted by the students so that I can give feedbacks as **comments**
Acceptance Criteria	• Ability to view all the biweekly reports in one place. • Ability to give feedbacks as **comments** to each the report.

User Story

Content	As a **user** I should be able to view the notifications to keep me updated with the actions taking place in the application.
Acceptance Criteria	• There should be notification if a new **proposal/idea** is added. • There should be notification if any change is made in an existing **proposal** like o **Proposal status** is changed. o New contributor requests to join an existing **proposal**. o New **comments** are added. o New files are uploaded. • There should be notification if any deadline is approaching.

User Story

Content	As a **user - supervisor** I should have an option to select how frequently I want to receive email notification so that I am not bothered by the system generated emails all the time.
Acceptance Criteria	• Ability to choose the frequency of the days after which I wish to receive an email. • Ability to receive consolidated emails.

User Story

Content	As a **user - student** I should receive an email notification of all the actions taking place in the application to keep me updated.
Acceptance Criteria	• There should be notification regarding **status** change. • There should be notification for new **comments** added. • There should be notification for new contributor request. • There should be notification for new files added. • There should be notification for new **proposal**/**idea** added. • Ability to stop receiving email notification

User Story

Content	As a **user** I should be able to change my password if I forget the earlier password.
Acceptance Criteria	• Ability to set new password. • An email should be sent if I want to change my password to confirm my user type.

User Story

Content	As a **user-student** I should be able to edit my profile settings so that I can keep my data updated.
Acceptance Criteria	• Ability to change matriculation number. • Ability to change the current semester. • Ability to change the beginning year.

User Story

Content	As a **user-student** working on a **project/thesis** I should be able to accept request from other students to join my project/thesis.
Acceptance Criteria	• Ability to view the contributors in an existing proposal. • Ability to see if new students requested to join the project/thesis. • Ability to approve/reject student request to join the project/thesis.

User Story

Content	As a **user-supervisor** I should have the right to accept new student request to join an existing proposal.
Acceptance Criteria	• Ability to view the contributors in an existing proposal. • Ability to approve or reject student request to join an existing proposal.

User Story

Content	As a **user-student** I should be able to submit my **project/thesis** to the professor after completion, for final review and grading.
Acceptance Criteria	• Ability to upload documents to the system. • Ability to choose which documents to be sent to the professor. • Ability to submit the project/thesis to the professor for final review.

User Story

Content	As a **user-supervisor** I should be able to review the final documents submitted by the students so that I can submit it to the Admin office.
Acceptance Criteria	• Ability to view all the files submitted by the **student**. • There should be provision for the professor to set the percentage of the requirements met by the student based on the initial proposal.

- Ability to choose if the **project/thesis** was successfully completed or was rejected.
- Ability to send a report to the admin office.

7 Other Nonfunctional Requirements

7.1 Performance Requirements

The FH Kiel Ticketing App is going to be web-based. Therefore it requires a powerful server with high bandwidth in order to handle heavy data flow. This also includes the capabilities of the server to handle a large number of students simultaneously.

Along with a powerful CPU the server must have the capabilities to handle the following technologies:

- ASP.Net
- ASP.Net MVC
- MS SQL

When designing the website should be able to load on the latest browsers such as Google Chrome, Opera, Mozilla Firefox etc.

An internet connection is necessary for the website to be used.

7.2 Security Requirements

Since the FH Kiel Ticketing App will be hosted on a server along with a database, the product chosen to do so must have the features to keep the data secure and prevent any kind of loss or harm to personal data.

A **user** can only log in the system through their own registered credentials and shall not be able to access data of another **user.**

A **user** shall only be able to access his/her own workspace i.e. the workspace provided by the product and shall not be able to access the workspace that is provided by the hosting server.

8 GitHub Repository Link

https://github.com/adnanchang/fh-kiel-ticketing-app.git

9 References

[1] C. Henderson, *Building Scalable Web Sites*. O'Reilly Media, Inc., 2006.
[2] D. Bertram, "The Social Nature of Issue Tracking in Software Engineering," p. 144.
[3] J. Janák, "Issue Tracking Systems," p. 106.
[4] "Zendesk | Customer Service Software & Support Ticket System," *Zendesk*. [Online]. Available: https://www.zendesk.com.
[5] "Online Help Desk Software | Self Service Knowledge Base | Community Forum." [Online]. Available: https://www.happyfox.com/ticket-support-system/..
[6] "Helpdesk Ticketing System | Get Started for Free." [Online]. Available: https://www.freshworks.com/content/en-US/freshdesk/helpdesk-software/.

Appendix A: Glossary

In this Glossary you will find definitions and context clarifications. Throughout the document there are instances where one must have a clearer picture of the concerned topic in order to understand the main objective of what is being said. This glossary caters to that and explains everything for a better understanding.

Definitions

There are certain words in the document that have a different definition within the context of the document. To provide a clearer perspective on what these words actually point to, these words are in **bold form** throughout the document which can be found chronologically here:

Idea(s): An Idea is an entity of the system which holds information regarding the content of the student's main goal. To elaborate further, an idea could be for a Master Project or a Thesis. The Idea will contain a short description when a Professor would create it. The Idea will contain the same information plus a Proposal when a Student would create it.

Ticket(s): A Ticket is an entity that is the center of attention in the system. The Ticket holds information regarding the members in it and what information have these members exchanged. When a Student wants to work with a Professor regarding his/her Project/Thesis a Ticket would be a place where all the concerned members (Students and Professors) can exchange information in the form of comments and files.

User(s): A user is an entity that refers to two types of main user classes:

- Student
- Supervisor

Whenever the document mentions the user classes as a "user" that means that both user classes have the ability to do whatever is stated or both will be affected with whatever is stated.

Administrator(s): An administrator is a system user or a user that can make changes to the system's data.

Field(s): A field is a category that defines the specialization of a student. These categories are pre-defined by the University for the Programs. For example, in the Masters in Information Engineering Program there are four specialization category: IT – Security, Business Management, Intelligent Systems, and System Development. So in the context of the system these four are "fields".

Role(s): Since the system is supposed to be made scalable so that the design can be used for other purposes as well, the concept of **roles** has been introduced in the system. This allows the system to be able to identify the type of the **user** trying to register/login.

Role Identifier(s): Role identifier is basically the detail to a **role.** For example, a **User** of type Student has the **role** of a student and is identified by "student.fh-kiel.de" in his/her email. Therefore "student.fh-kiel.de" is the **Role Identifier** for the **Role** Student.

Status: The status of the ticket shows the current situation of the **ticket.** To elaborate, the status would provide the **user** with information regarding the current position of the **ticket** i.e. whether the **ticket** is in progress, or the perhaps the **proposal** needs adjustments or the **ticket** is closed permanently. The statuses are to be decided in the final product and are not yet finalized.

Comment: A comment in the system is the way the members of a **ticket** will communicate with each other.

Context Clarification

There are certain words in the document that may point to ambiguity. To eliminate such cases here are the definitions of the words followed with a superscripted number $^{[1]}$ throughout the document to explain the context and need behind it.

[a] *mainly*: The document can be read by a lot of people, maybe even new students who would like to undertake the project for further development but as of the current needs, the document will be read by the people listed.

[b] *In the future*: This document is made to illustrate the workings of a software that is a master project of a group of people. This is, however, not the final version. There are possibilities that some other functionalities maybe added. All these possible functionalities are not covered in the document.

[c] *Data*: The requirement states that the user can enter their data in order to register. The data here means the general information one has to provide in order to register their account i.e. First Name, Last name, email, and password.

[d] *Final Files*: Whenever a student starts a project or a thesis there are some files that he/she must submit at the end of it in order for the project/thesis to be called complete. These files are not always pre-defined. These files could be a document or a piece of code. This depends on the discussion that is done between the Professor and the Students.

[e] *Adequate Data:* Whenever a thesis begins, the supervisor that is looking into the thesis has to fill in a form and email it to the examination department. This form contains the name of the student doing the thesis, the professor supervising the thesis, and the topic of the thesis along with start and end date.

Appendix B: Analysis Models

Data Diagram

Following is the currently designed data diagram and the description of all tables.

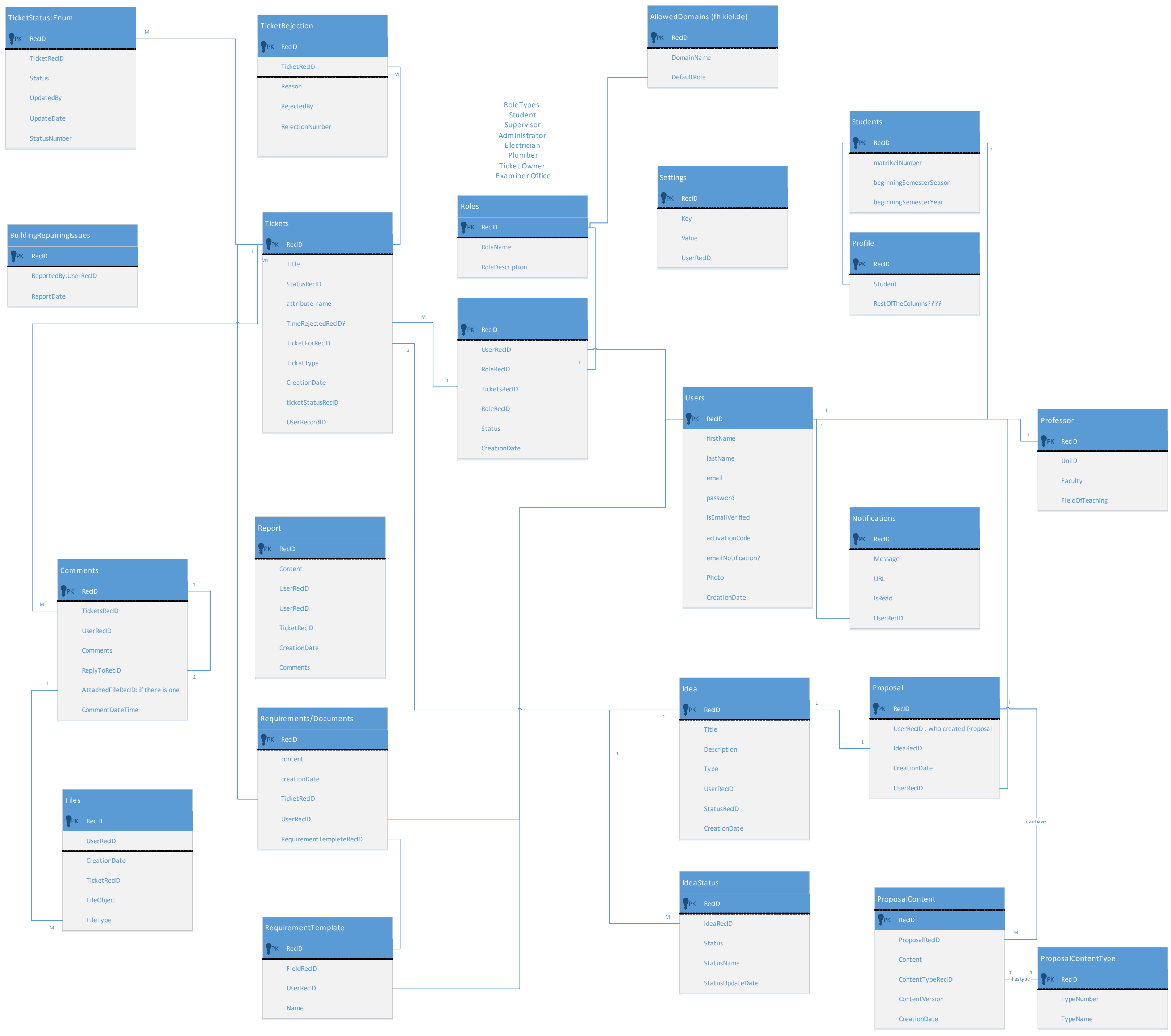


Table Name:	*Ticket*
Description	This table will hold all the **tickets** that will be generated in the system. The design of the table has kept in mind the scalability aspects of the system not forgetting that the design might/shall be used for other purposes in the future apart from keeping track of the projects and thesis of students at FH Kiel. WRITE MORE
Relationships	• **Has primary key:** RecID • **Has foreign keys from**: ○ Users ○ TicketFor • **Is a foreign key in**: ○ Comments ○ Requirements ○ TicketRejection ○ Contributors ○ TicketStatus ○ Idea

Table Name:	*TicketRejection*
Description	TicketRejection table keeps track and history of the tickets that have been rejected by **user – supervisors.** It stores information about the reason of rejection and the date and time of the rejection. It will also help denote the number of times a certain ticket has been rejected.
Relationships	• **Has primary key:** RecID • **Has foreign keys from**: ○ Ticket • **Is a foreign key in**: ○ Is not a foreign key in any table
Remarks	

Table Name:	*TicketStatus*
Description	Whenever the **status** of a **ticket** has been changed, the TicketStatus table stores that entry and also keeps track of the previous **statuses** of a certain **ticket.**
Relationships	• **Has primary key:** RecID • **Has foreign keys from**: ○ Ticket

	• **Is a foreign key in**: o Is not a foreign key in any table
Remarks	

Table Name:	*Users*
Description	The User table contains information about any individual that will use the system. This table acts as a superclass to all the other **users** that have different roles in the system. An example in this case would be the student and the supervisor who are in the base level stored in the User table.

Relationships	• **Has primary key:** RecID
	• **Has foreign keys from**:
	  o Does not have any
	
	• **Is a foreign key in**:
	  o Comments
	  o Requirements
	  o Contributors
	  o Idea
	  o Proposal
	  o Supervisor
	  o Student
	  o Requirements
	  o RequirementTemplate
	  o Files
	  o Comments

Table Name:	*Profile*
Description	The Profile table keeps information about **user – students.** This table will help the system extract information about a certain student in order to conclude whether or not the student should be a part of a **ticket** or not.
Relationships	• **Has primary key:** RecID • **Has foreign keys from**: o Students • **Is a foreign key in**: o Is not a foreign key in any table.

Remarks	Has a design flaw, the field **Ticket.TicketForRecID** is a foreign key for this table and other table (shown in diagram) at the same time.

Table Name:	*Contributors*
Description	A **ticket** can have many members. In other words, one can say that these members are contributing to this **ticket.** The Contributors table stores information about all the members of a certain **ticket.** It also stores information about their roles and rights in a **ticket.**
Relationships	• **Has primary key:** RecID • **Has foreign keys from**: o Roles o Users o Tickets • **Is a foreign key in**: o Is not a foreign Key in any table

Table Name:	*Supervisor*
Description	The supervisor table contains information about the Professors. It is a child table to the User table.
Relationships	• **Has primary key:** RecID • **Has foreign keys from**: o Users • **Is a foreign key in**: o Is not a foreign key in any table.

Table Name:	*Fields*
Description	A field is a category that defines the specialization of a student. These categories are pre-defined by the University for the Programs. For example, in the Masters in Information Engineering Program there are four specialization categories: IT – Security, Business Management, Intelligent Systems, and System Development. So in the context of the system these four are “fields”.
Relationships	• **Has primary key:** RecID • **Has foreign keys from**: o User • **Is a foreign key in**:

	○ Is not a foreign key in any table
Remarks	

Table Name:	*Student*
Description	The student table contains information about the students. It is a child table to the User table.
Relationships	• **Has primary key:** RecID • **Has foreign keys from**: ○ Users • **Is a foreign key in**: ○ Profile

Table Name:	*Roles*
Description	Since the system is supposed to be made scalable so that the design can be used for other purposes as well, the concept of **roles** has been introduced in the system. This allows the system to be able to identify the type of the **user** trying to register/login. It also tells us about the roles of a **contributor** in a **ticket.**
Relationships	• **Has primary key:** RecID • **Has foreign keys from**: ○ Users ○ TicketFor • **Is a foreign key in**: ○ Contributors ○ Users

Table Name:	*Idea*
Description	An Idea is an entity of the system which holds information regarding the content of the student's main goal. To elaborate further, an idea could be for a Master Project or a Thesis. The Idea will contain a short description when a Professor would create it. The Idea will contain the same information plus a Proposal when a Student would create it.
Relationships	• **Has primary key:** RecID • **Has foreign keys from**: ○ Users ○ Ticket

	• **Is a foreign key in**: ○ Proposal
Remarks	Has a design flaw, the field **Ticket.TicketForRecID** is a foreign key for this table and other table (shown in diagram) at the same time.

Table Name:	*IdeaStatus*
Description	For an **idea** created by a **user – supervisor,** it is available for all students. When two or more students separately choose the **idea** by writing a proposal, the IdeaStatus table tells us about the current circumstance. For example, in this case the status would be in a *bidding* mode. Whereas when a student's proposal is selected the status would be *closed* i.e. not available anymore.
Relationships	• **Has primary key:** RecID • **Has foreign keys from**: ○ Idea • **Is a foreign key in**: ○ Is not a foreign key in any table
Remarks	

Table Name:	*Proposal*
Description	The Proposal table holds information about a proposal that a student would write to raise a **ticket.** This table would also show the history of all previous proposals written for a **ticket.**
Relationships	• **Has primary key:** RecID • **Has foreign keys from**: ○ Idea ○ Users • **Is a foreign key in**: ○ Proposal Content
Remarks	

Table Name:	*ProposalContent*
Description	This table holds information about a certain proposal. This table keeps history of the content that has been changed for a proposal.
Relationships	• **Has primary key:** RecID

	• **Has foreign keys from**: ○ Proposal • **Is a foreign key in**: ○ ProposalContentType
Remarks	

Table Name:	*ProposalContentType*
Description	Inside the proposal one has different types of contents. This table helps the student identify the content that he/she is writing belongs to which type.
Relationships	• **Has primary key:** RecID • **Has foreign keys from**: ○ Has no foreign key • **Is a foreign key in**: ○ Is not a foreign key in any table
Remarks	

Table Name:	*TicketStatus*
Description	The status of the **ticket** shows the current situation of the **ticket.** To elaborate, the status would provide the **user** with information regarding the current position of the **ticket** i.e. whether the **ticket** is in progress, or perhaps the **proposal** needs adjustments or the **ticket** is closed permanently.
Relationships	• **Has primary key:** RecID • **Has foreign keys from**: ○ Has no foreign key • **Is a foreign key in**: ○ Tickets
Remarks	

Table Name:	*Comments*
Description	A comment in the system is the way the members of a **ticket** will communicate with each other.

Relationships	• **Has primary key:** RecID
	• **Has foreign keys from**:
	  o Users
	  o Ticket
	  o Files
	  o Comments (self)
	• **Is a foreign key in**:
	  o Comments (self)

Remarks	

Table Name:	*Files*
Description	This table holds all files uploaded for a **ticket.**

Relationships	• **Has primary key:** RecID
	• **Has foreign keys from**:
	  o Users
	  o Ticket
	• **Is a foreign key in**:
	  o Comments

Remarks	

Table Name:	*Requirements*
Description	A **ticket** will have a certain requirement in order for it to be considered complete. This table will hold data about these requirements. For example, by the end of a **ticket** a certain number of files must be submitted.
Relationships	• **Has primary key:** RecID • **Has foreign keys from**: o Users o Ticket o Field • **Is a foreign key in**: o Is not a foreign key in any table
Remarks	

Table Name:	*AllowedDomains*

Description

AllowedDomains is basically the detail to a **role.** For example, a **User** of type Student has the **role** of a student and is identified by “student.fh-kiel.de” in his/her email. Therefore “student.fh-kiel.de” is the Role Identifier for the **Role** Student thus allowing that domain.

Relationships

- **Has primary key:** RecID
- **Has foreign keys from**:
    - Roles
- **Is a foreign key in**:
    - Is not a foreign key in any table

Remarks

Table Name: *RequirementsTemplate*

Description

In order to describe the final requirements of a **ticket** usually these requirements don’t change and remain the same. To facilitate the supervisor, they can use this table to store a template that will act as the final requirement to a **ticket.**

Relationships

- **Has primary key:** RecID
- **Has foreign keys from**:
    - Has no foreign key
- **Is a foreign key in**:
    - Requirements

Remarks

Table Name: *Notifications*

Description

This table holds all notifications sent to all **users.**

Relationships

- **Has primary key:** RecID
- **Has foreign keys from**:
    - User
- **Is a foreign key in**:
    - Is not a foreign key in any table

Remarks

Table Name: *Settings*

Description

Contains Parameters for user preferences

Relationships

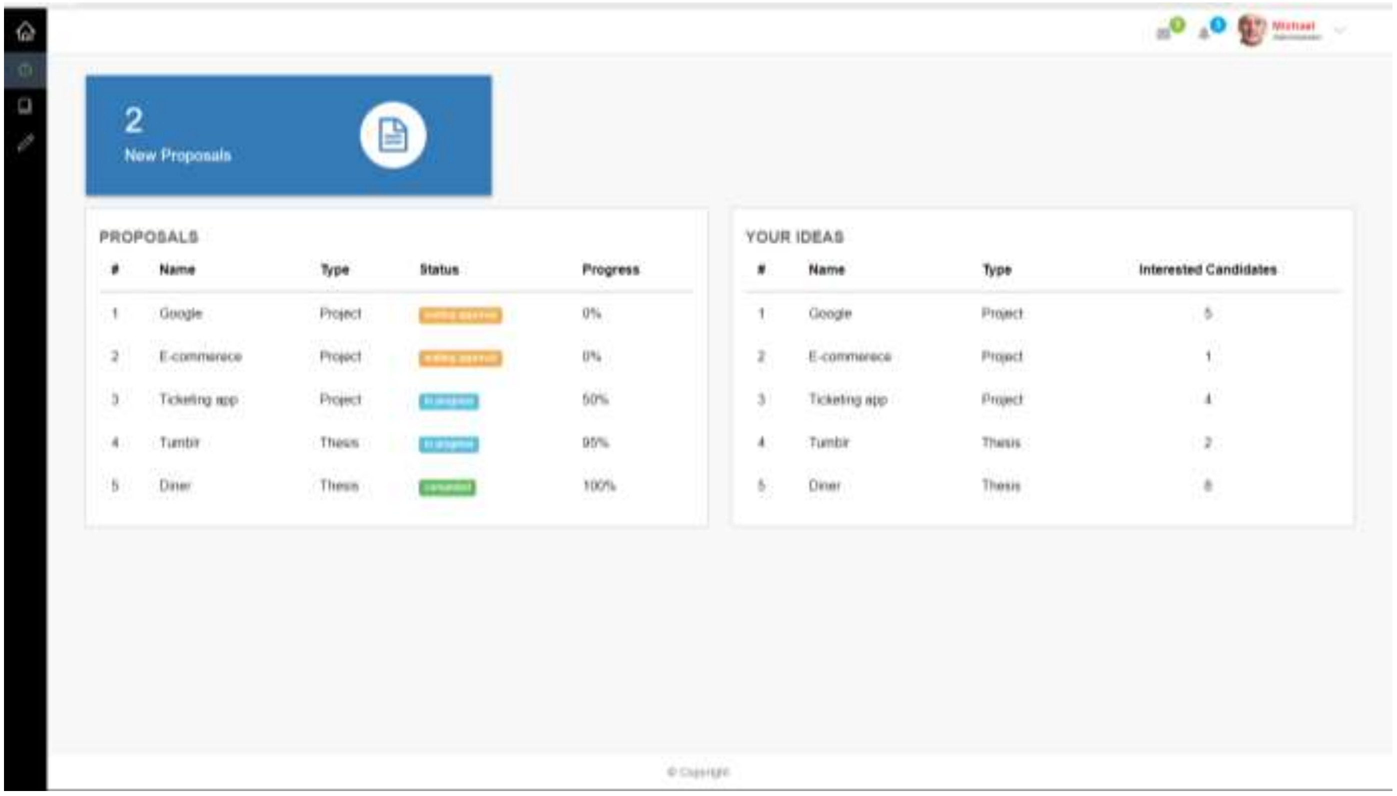
- **Has primary key:** RecID
- **Has foreign keys from**:
    - User

	• **Is a foreign key in**: ○ Is not a foreign key in any table
Remarks	

Table Name:	*BiweeklyReport*
Description	This table will contain a biweekly report for a **ticket.** These reports are produced by the student in order to keep the supervisor up to date with the progress of the **ticket.**
Relationships	• **Has primary key:** RecID • **Has foreign keys from**: ○ Ticket • **Is a foreign key in**: ○ Is not a foreign key in any table
Remarks	

Appendix C: Design Prototype

Following are mockups providing a more vivid look at what the final design shall look like:

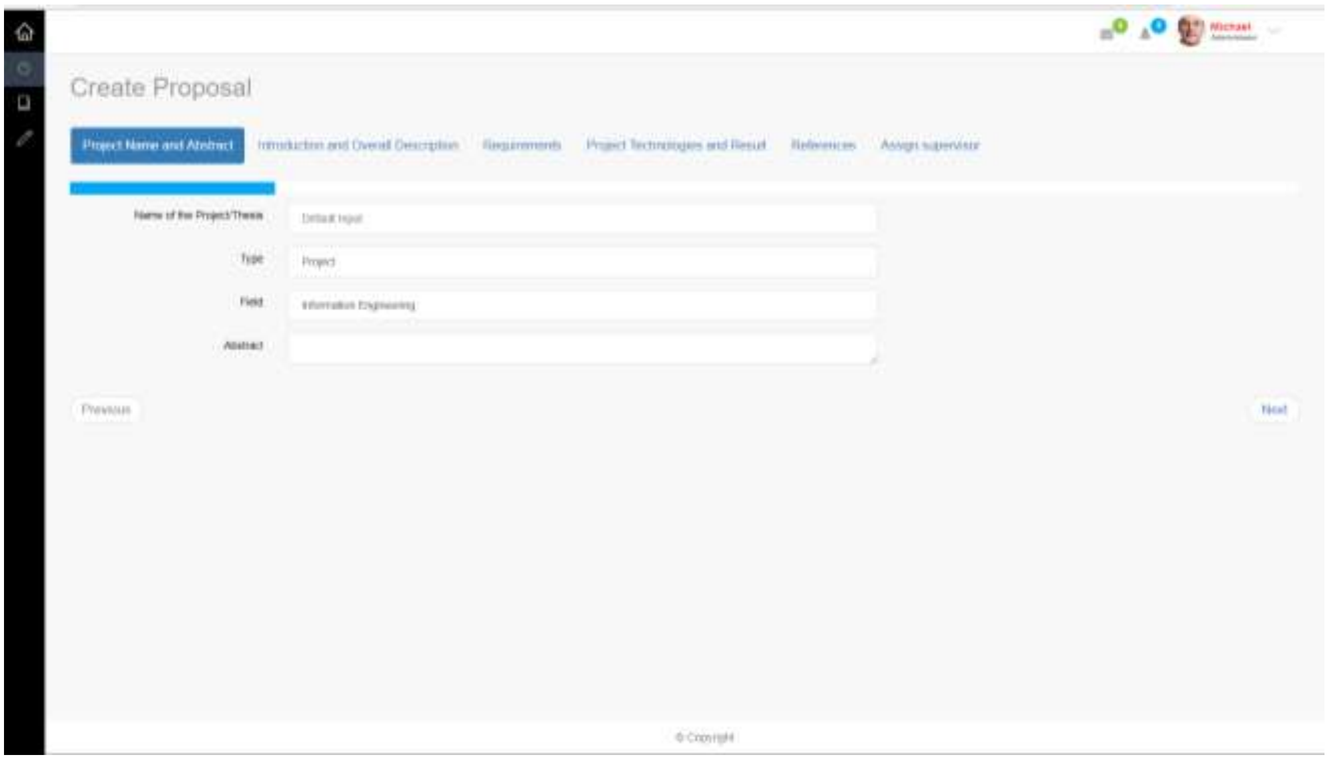


PROPOSALS

#	Name	Type	Status	Progress
1	Google	Project	[illegible]	0%
2	E-commerece	Project	[illegible]	0%
3	Ticketing app	Project	[illegible]	50%
4	Tumblr	Thesis	[illegible]	95%
5	Diner	Thesis	[illegible]	100%

YOUR IDEAS

#	Name	Type	Interested Candidates
1	Google	Project	5
2	E-commerece	Project	1
3	Ticketing app	Project	4
4	Tumblr	Thesis	2
5	Diner	Thesis	8



Michael

Create Proposal

Project Name and Abstract | Introduction and Overall Description | Requirements | Project Technologies and Result | References | Assign supervisor

Name of the Project/Thesis: Default input

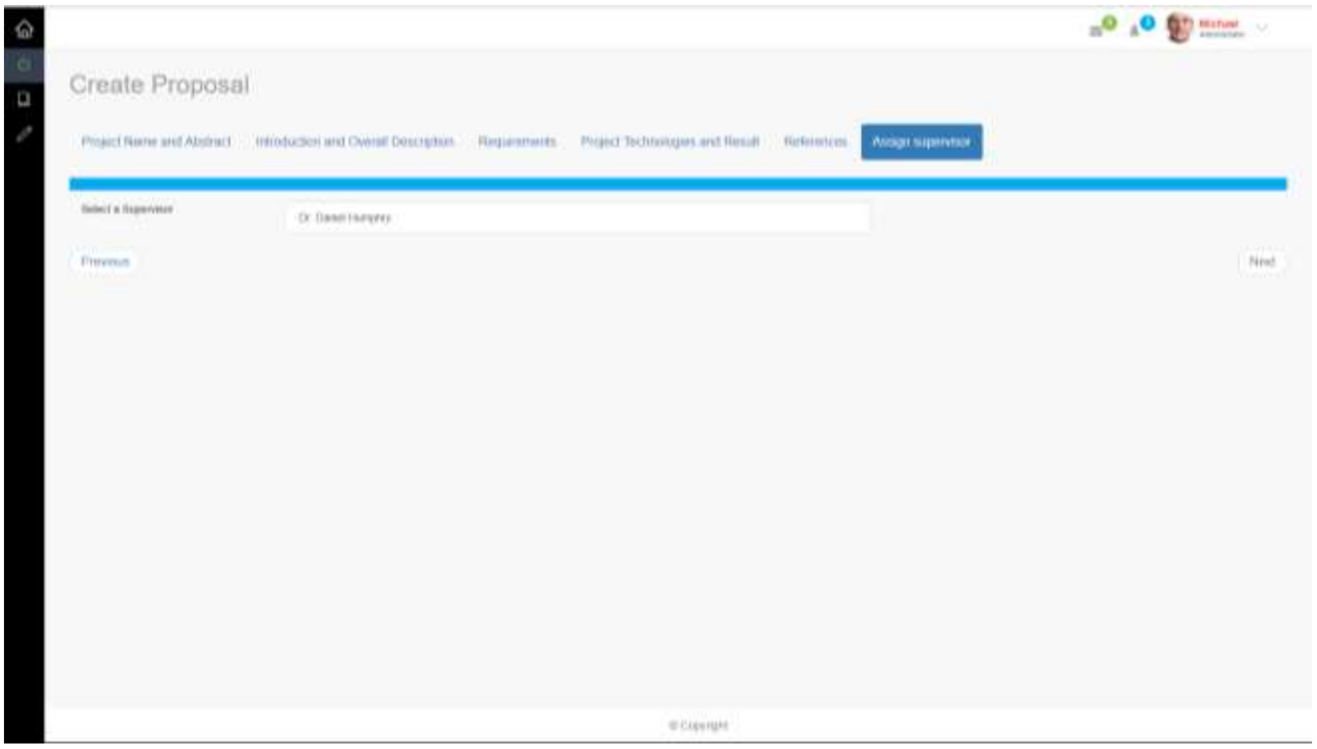
Type: Project

Field: Information Engineering

Abstract:

Previous | Next

© Copyright



Michael

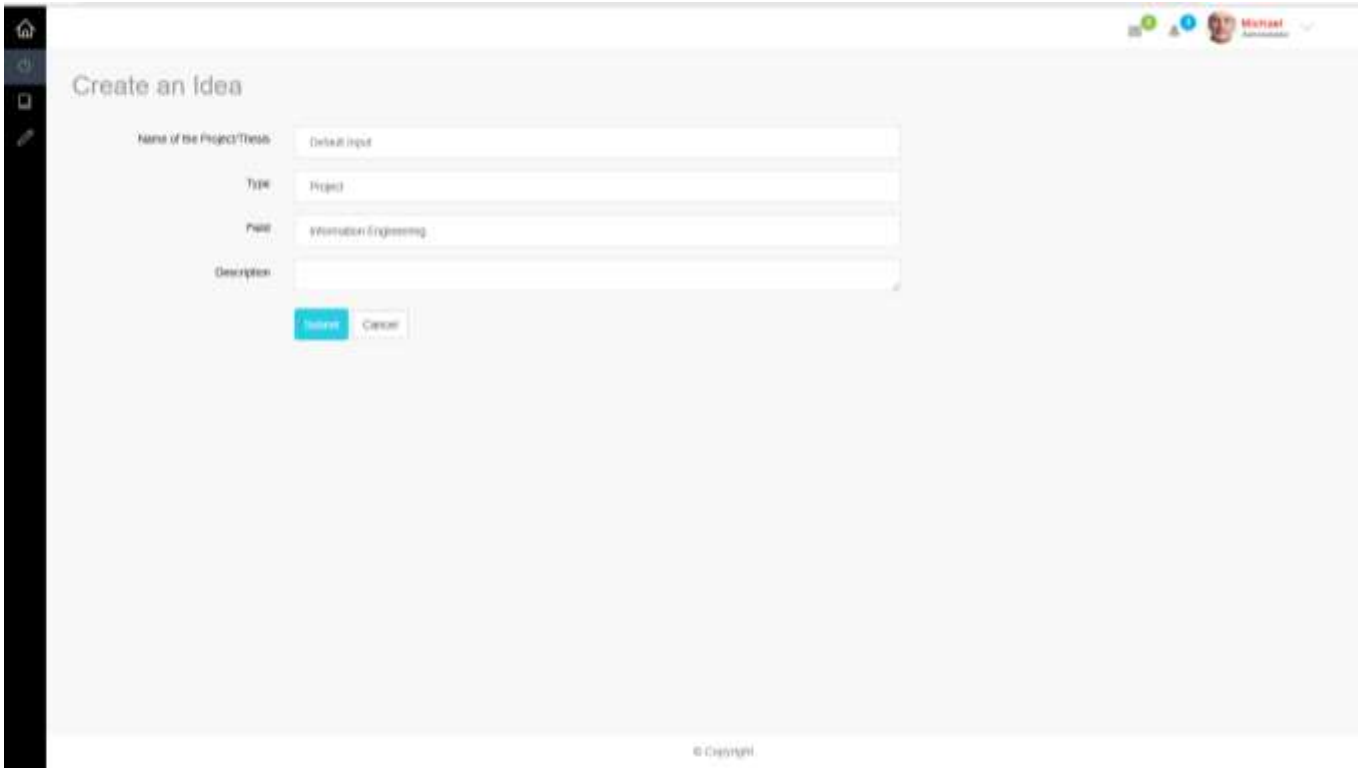
Create Proposal

Project Name and Abstract | Introduction and Overall Description | Requirements | Project Technologies and Result | References | Assign supervisor

Select a Supervisor: Dr. Daniel Humphry

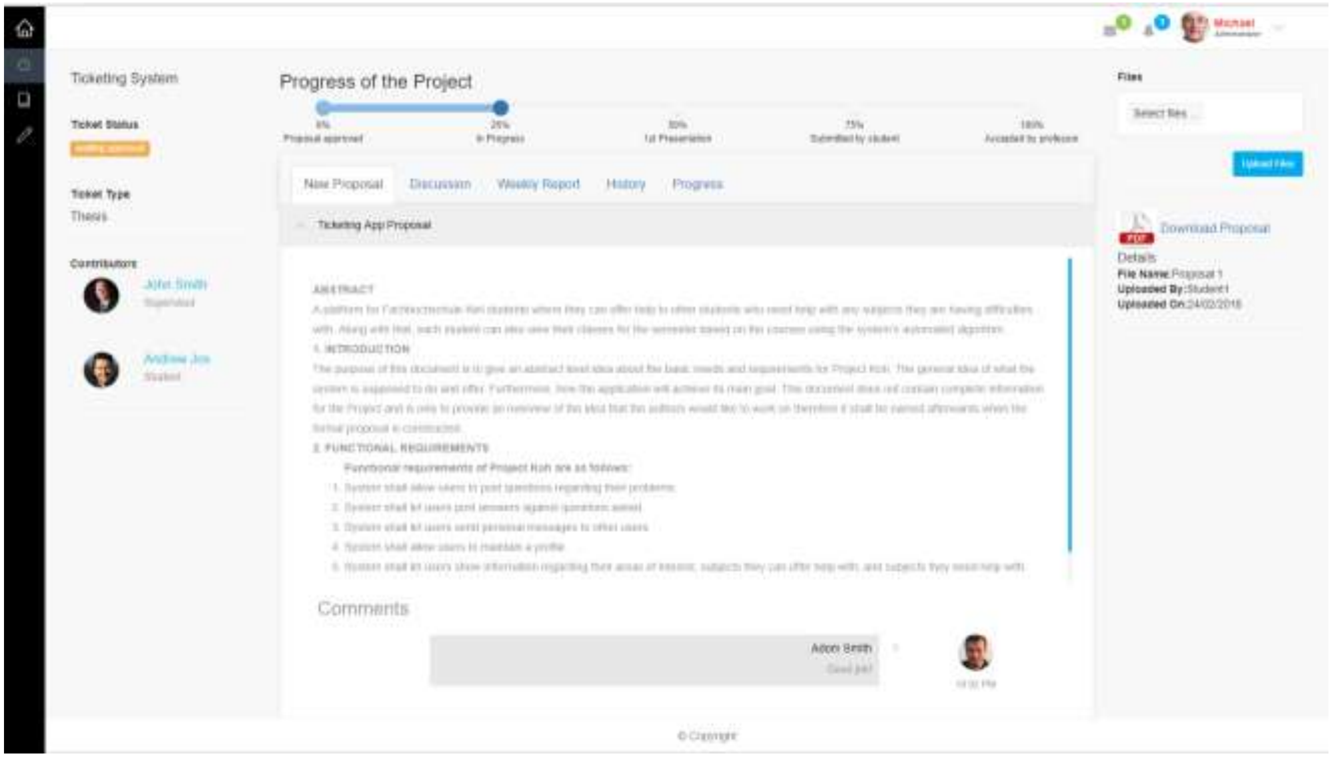
Previous | Next

© Copyright



Create an Idea

Name of the Project/Thesis	Default input
Type	Project
Field	Information Engineering
Description	



Ticketing System

Ticket Status
waiting approval

Ticket Type
Thesis

Contributors
John Smith
Supervisor
Andrew Jos
Student

Progress of the Project

0% Proposal approved
25% In Progress
50% 1st Presentation
75% Submitted by student
100% Accepted by professor

Ticketing App Proposal

ABSTRACT
A platform for Fachhochschule Kiel students where they can offer help to other students who need help with any subjects they are having difficulties with. Along with that, each student can also view their classes for the semester based on the courses using the system's automated algorithm.
1. INTRODUCTION
The purpose of this document is to give an abstract level idea about the basic needs and requirements for Project Koh. The general idea of what the system is supposed to do and offer. Furthermore, how the application will achieve its main goal. This document does not contain complete information for the Project and is only to provide an overview of the idea that the authors would like to work on therefore it shall be named afterwards when the formal proposal is constructed.
2. FUNCTIONAL REQUIREMENTS
Functional requirements of Project Koh are as follows:
1. System shall allow users to post questions regarding their problems.
2. System shall let users post answers against questions asked.
3. System shall let users send personal messages to other users.
4. System shall allow users to maintain a profile.
5. System shall let users show information regarding their areas of interest, subjects they can offer help with, and subjects they need help with.

Comments

Adom Smith
Good job!
12:32 PM

Files
Select files ...

Details
File Name: Proposal 1
Uploaded By: Student1
Uploaded On: 24/02/2018