TECHNISCHE UNIVERSITÄT
ILMENAU

IMMS
INSTITUT FÜR MIKROELEKTRONIK- UND MECHATRONIK-SYSTEME

Internship Report

Research in Computer and System Engineering
Master Program

Pablo Johnson Rojas
Matriculation Number: 62662

Company Supervisor:

Georg Gläser
Georg.Glaeser@imms.de

Content Index

Company Information

The Institute for microelectronic and mechatronic systems (IMMS) was founded in 1995 as a state-owned company of the Free State of Thuringia and the Ilmenau University of Technology institute. It currently has about 80 people working between the headquarters in Ilmenau and Erfurt.

The IMMS is aimed to build a bridge between science and industry and continuously strengthen it through close cooperation with universities such as the Technical University of Ilmenau, with industry partners, involvement in networks and competence clusters, and consistent support for young researchers. It supervises up to 40 students a year and enables them with sound methodological knowledge and involvement in practice for a career start in the industry and application-oriented research. One of the main objectives of the IMMS is to strengthen small and medium-sized enterprises, in particular with application-oriented research and development in microelectronics, systems engineering, and mechatronics, and to transfer the results of basic research into applications and products.

As a strategic partner, it supports companies in launching internationally successful innovations for health, the environment, and industry and accompanies them from the feasibility study to series production. This support can be costly and personnel-intensive over several years and cannot usually be carried out by pure basic research at universities. SMEs, in particular, are generally unable to carry out such a transfer on their own. As a rule, they lack the research capacities and financial potential.

For this purpose, the IMMS develops and optimizes system components, assemblies, circuits, and communication between all system elements and between system and environment. The IMMS researches integrated sensor systems, intelligent networked measurement and test systems, and magnetic 6D direct drives with nanometer precision. The IMMS opens up high-performance solutions for digitization and miniaturization with application-oriented developments, which are the key to new applications.

Integration into the company

Georg, my supervisor, and the team received me the best way. I was introduced to the entire team of designers, PCB analysts, interns, and company staff on the first day. I was also given a tour of the different areas and offices and was informed about the diverse projects being carried out. I could see that the workspace is a very horizontal space where all the workers interact equally. On this day, I was also given my workplace and a computer with the necessary resources, such as the email account, network user, and servers configuration.

Due to the current COVID19 pandemic situation, the operational mode was mixed, going once or twice a week to the office and the other days working from home. Despite this, I was able to get to know most of my colleagues at IMMS-Erfurt and enjoy pleasant moments during the hours I was in the office.

Additionally, we developed a methodology in which we have daily meetings between all trainees and supervisors. Each trainee mentions what was done the day before, what will be done this day, and if he/she has any problems or needs anything from anyone. These meetings allowed me to get to know the other interns and supervisors and learn about the different projects that are being developed in the institution and how to make my project known to others.

Other activities like seminars were also beneficial to sharing and learning the work from other interns. For example, I attended three different seminars and gave one to show my other colleagues my current work and progress.

.

The task

Printed circuit boards (PCBs) are a fundamental part of people's daily lives. They are essential parts of any everyday electronic device such as a TV, a computer, or a cell phone. Moreover, with the arrival of new technologies such as the Internet of Things (IoT), they have become even more integrated into people's lives. For these reasons, it is necessary to improve and accelerate the design and construction process to reduce production costs and support more and new technologies.

Our partner (a PCB manufacturer) has defined a PCB design and production process, in which once the design is ready, it is evaluated through different design checking rules (DCR). However, it is not possible to detect all potential problems in this review, so the result of the DCR must be evaluated manually through a visual check. In this observation, design structures that cannot be realized with the established manufacturing technologies are repeatedly observed. Therefore, these observations need to be recognized and corrected. However, at present, it is tough to automate this check since corrections would only be made according to the established parameters without considering the netlist or other parameters. For this reason, the visual inspection is done by an expert in the field, which means the use and dependence on specialized human resources and a potential bottleneck in production.

A recurring problem is mentioned below as a case study example:

> **Problem:** Insufficient copper exposure of non-plated holes (NPH).
>
> A copper-free area of at least 300µm around the NPHs is required as a contact surface for sufficient resist adhesion and to avoid galvanic copper build-up. The expert should individually evaluate areas where the distance is less than specified (Figure 1). If necessary, manually correct them since the distance is too small, the PCB may fail. In addition, resistance infiltration may occur, attacking and damaging nearby structures.

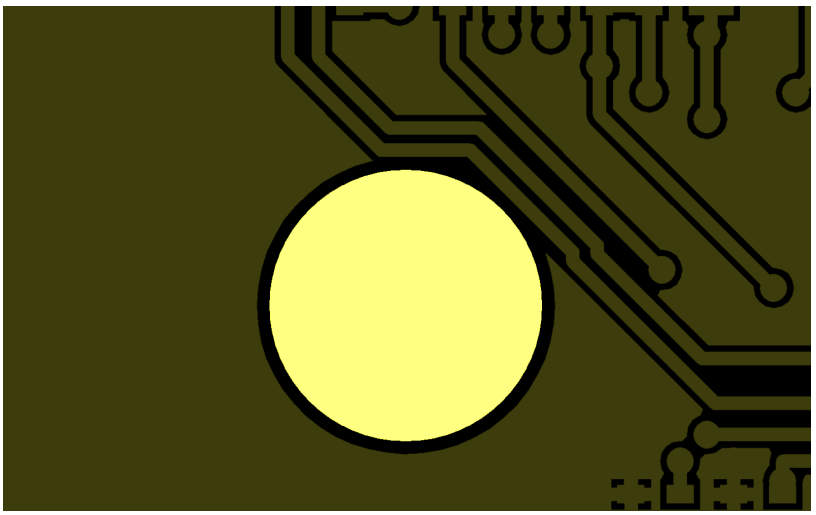

Figure 1: Example of insufficient exposure to copper of non-plated holes.

Solution: The solution is to remove the copper surfaces and conductors as far away as possible from the NPH holes to ensure the laminate adhesion.

This internship aims to create a software that provides the necessary interfaces to support the processes that will help optimize the manufacturing processes to ensure a smooth production.

We have identified two main processes that will be of great importance to support a machine learning-based model to optimize PCB manufacturing processes. The first process has been denominated as "Training," The potential errors for a later training of the model are identified by comparing an original PCB layout design with the corrected PCB layout design and tagging each finding with the correct error. The second process is the Inference process, in which the detection of possible errors in a PCB design is performed using the trained model.
It is worth mentioning that these processes will not be developed during the current internship; This internship is aimed only at creating a tool that supports them.
The following diagram (Figure 2) shows the flow of both processes in detail.

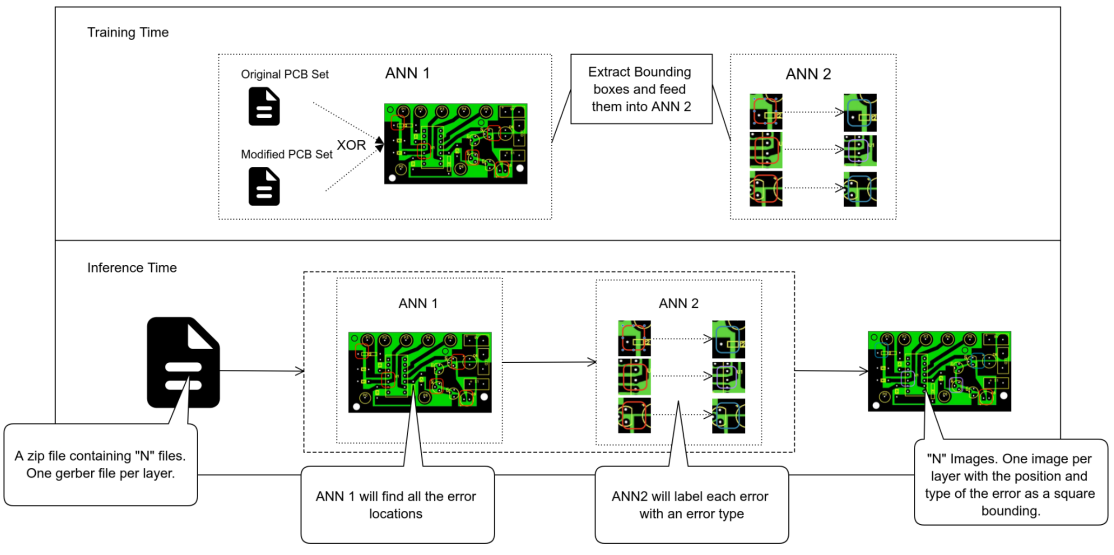

Figure 2: Diagram of the "Training" and "Inference" processes.

Problem solving

The solution consists of creating a graphical interface that supports the training and inference processes, and specific alternative flows transversal to the use of the application. Alternative flows that provide a more significant utility to the solution, such as user and error type management. The requirements for the different functionalities previously mentioned are detailed below:

1. Users management

    - The system must allow the creation of users for each person who wishes to interact with the system.
    - The system must allow the creation of two different types of users, superuser users and regular users. Superuser users will be able to create other users and create and modify error types in addition to all the functionalities of regular users.
    - The system must allow users to log in with their email and password.
    - The system must allow the user to modify his personal data such as name, email, username, and password.
    - The system must list all users created in the application.

2. Error types management

    - The system must allow the creation, modification, and list of error types.
    - The error types must contain a name, a description, and a reference image.
    - Error types must be visible to all users.
    - Error types can only be created and modified by users with a superuser role.
    - The deletion of an error type must be logical, and all errors cataloged with that error type must be invalidated.

3. Creation and tagging of errors (Training flow)

    - The system must allow generating from two PCB layouts (one original and one modified with the corrections) an image for each difference found in each layer of the PCB design.
    - The system must allow cataloging of each difference found with a type of error.
    - Each difference must belong to a layer of the layout design, and each layer must belong to a PCB design. This capable set will be called "Training."
    - Each "Training" will consist of a name, a description, and two .zip files. The first one will have the original PCB design, and the second one will contain the modified design with corrections.
    - The zips files can only contain Gerber files inside. Each Gerber file is equivalent to one layer of the design.
    - Once the training is created, the system will display a list of each difference found between the designs and allow each difference to be cataloged with an error type.
    - The system will allow changing the status of the training to *reviewed* once all differences have been cataloged.
    - The system will allow assigning a reviewer user to the training when it is in *reviewed* status, and the training will pass to the *assigned* status.
    - The system will allow the assigned user to review and correct the type of error assigned to a difference.
    - The system will allow the assigned user to approve or disapprove of the training.
    - The system will generate an image for each difference found. These images will then be used to create the error image dataset and train the respective models.

4. Detection of potential errors in a PCB design (Inference flow)

    - The system shall detect from a PCB layout potential errors.
    - The system will allow the creation of an "Inference," which will consist of a name, a description, and a zip file containing the PCB layout (Gerber files).
    - The system will display a list of potential errors separated by layers and cataloged with the types of errors that the model has inferred.
    - The system will allow the user to review and correct the error type assigned to a potential error.
    - The system will allow the user to change the status of the inference to reviewed.
    - The system will allow the system to assign a reviewer user to the inference once the inference is in the reviewed state, and the inference will go to the assigned state.
    - The system shall allow the assigned user to review and correct the type of error assigned to a potential error.
    - The system will allow the assigned user to approve or disapprove of the inference.
    - The system will generate an image for each error found. These images will then be added to the error image dataset.

Figure 3 below shows the state diagram of the Inference flow. The Training flow state diagram is omitted as it is very similar.

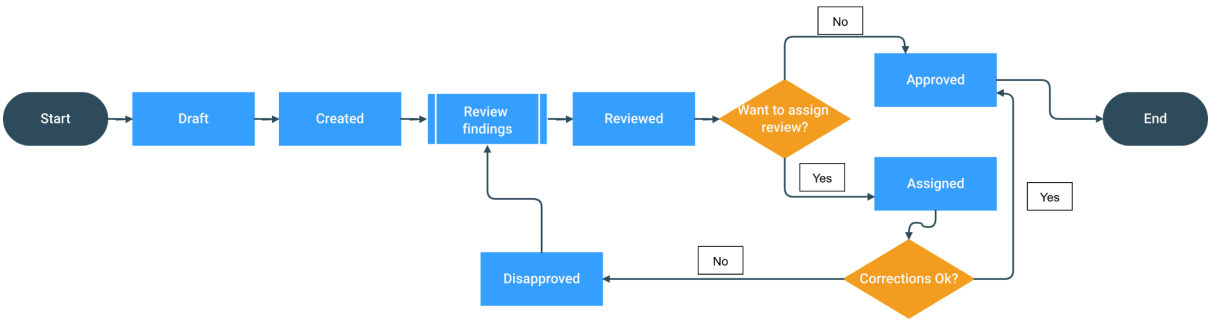

Figure 3: State diagram of the inference flow.

The solution will be a software capable of all the above mentioned, and that also meets specific non-functional requirements such as:

1. The software must be a desktop application.
2. The software must have a simple and user-friendly interface.
3. The software must be compatible with different operating systems such as Linux, macOS, and Windows.
4. The software must be easy to deploy.

We decided that the solution would have a client-server architecture model. Under this architecture, it is possible to have most of the business logic on the server and take advantage of the hardware features to execute the most intensive tasks in terms of processing and memory. Another important reason for this decision was to minimize the number of updates on the client in the event of a change in the business logic. In this way, the client will be responsible only for getting the inputs from the user and rendering the response from the server. As a consequence of this decision, we decided to handle the solution as two different projects. The client was called "Frontend," and the server was called "Backend." Following, we describe in detail the stack used for the development of each project as well as the libraries, frameworks, tools, and different decisions that were taken in both.

Backend

The language chosen for the development of this project was Python because, besides being one of the most popular languages, having a large community of programmers, and having an enormous number of libraries and tools that allow a better and faster development, Python is the predominant language within the IMMS team.

Based on the election of Python as the primary language for the backend, we had to choose which framework we would use as the web framework. We had mainly three options: to use

Django, Flask, or the novel FastAPI. Django and Flask are very known frameworks with solid bases and many options for integrating different features. On the other hand, FastAPI is a relatively young web framework (+3 years) but very intuitive, easy to code, robust, and with automatic interactive documentation out of the box (OpenAPI). Therefore, we decided to use FastAPI as the webserver framework considering the aforementioned features.

We decided to use PostgreSQL as the database system for the data persistency because we wanted to have a relational database. In addition, PostgreSQL is known for its good performance, extensive adoption in the community, and flexibility. On top of that, we decided to use SQLAlchemy as the object-relational mapping (ORM) tool for abstracting all the specific SQL logic and not depend on an exclusive database system. Eventually, we chose to use Alembic for managing the database migrations due to its great compenetration with SQLAlchemy.

Figure 4 shows the Entity-Relation diagram that will support all the business logic mentioned above.

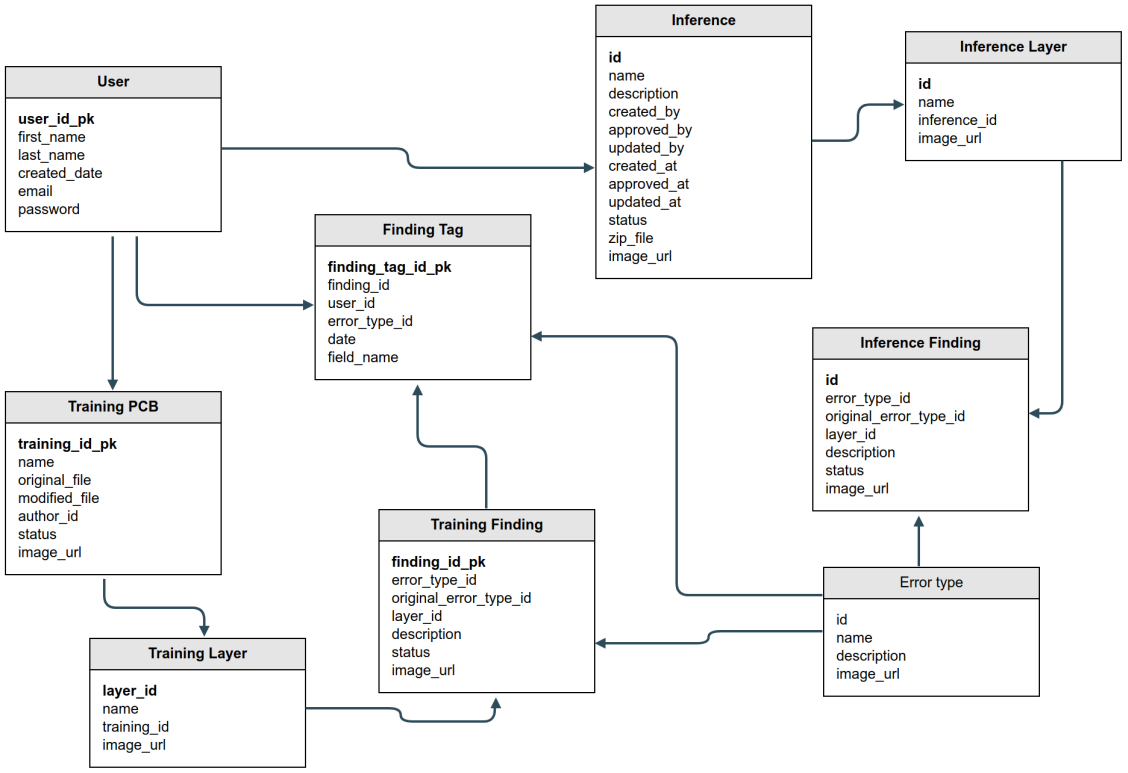

Figure 4: Entity-Relation diagram of the backend database.

Finally, to accomplish the non-functional requirements number 3 and 4, we decided to use Docker and Docker-Compose to create a backend compatible with any operating system and convert the deployment into a simple task.

Frontend

The language chosen for developing the frontend was javascript combined with Vuejs, the framework for building the web user interfaces. Of course, we had more options like react or angular but having other projects built on Vuejs was a powerful reason we chose it.
Vuejs ecosystem came with Pinia as the state manager and Vue-router as the one in charge of managing the navigation within the app.

We relied on the PrimeVue library and reused all its already validated components and patterns to accomplish the non-functional requirement number 2, a user-friendly interface.
In the case of the non-functional requirement number 1, we decided to use electronJs; in this way, we could develop a cross-platform desktop app with just one source code.

Figure 5, figure 6, and figure 7 show some of the initial frontend wireframes. These wireframes were created during the initial part of the internship; they served as a guide for the final user interface.

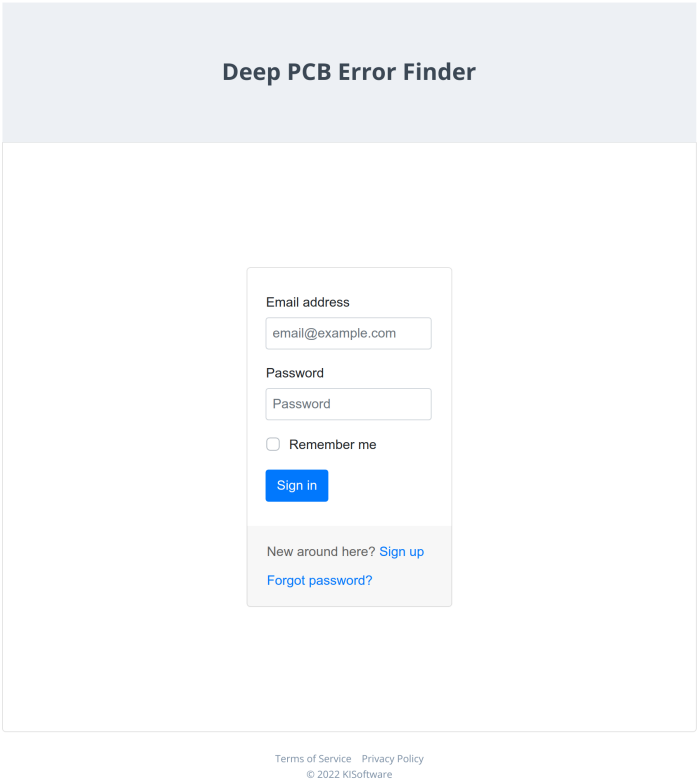

Figure 5: Wireframe of the login page.

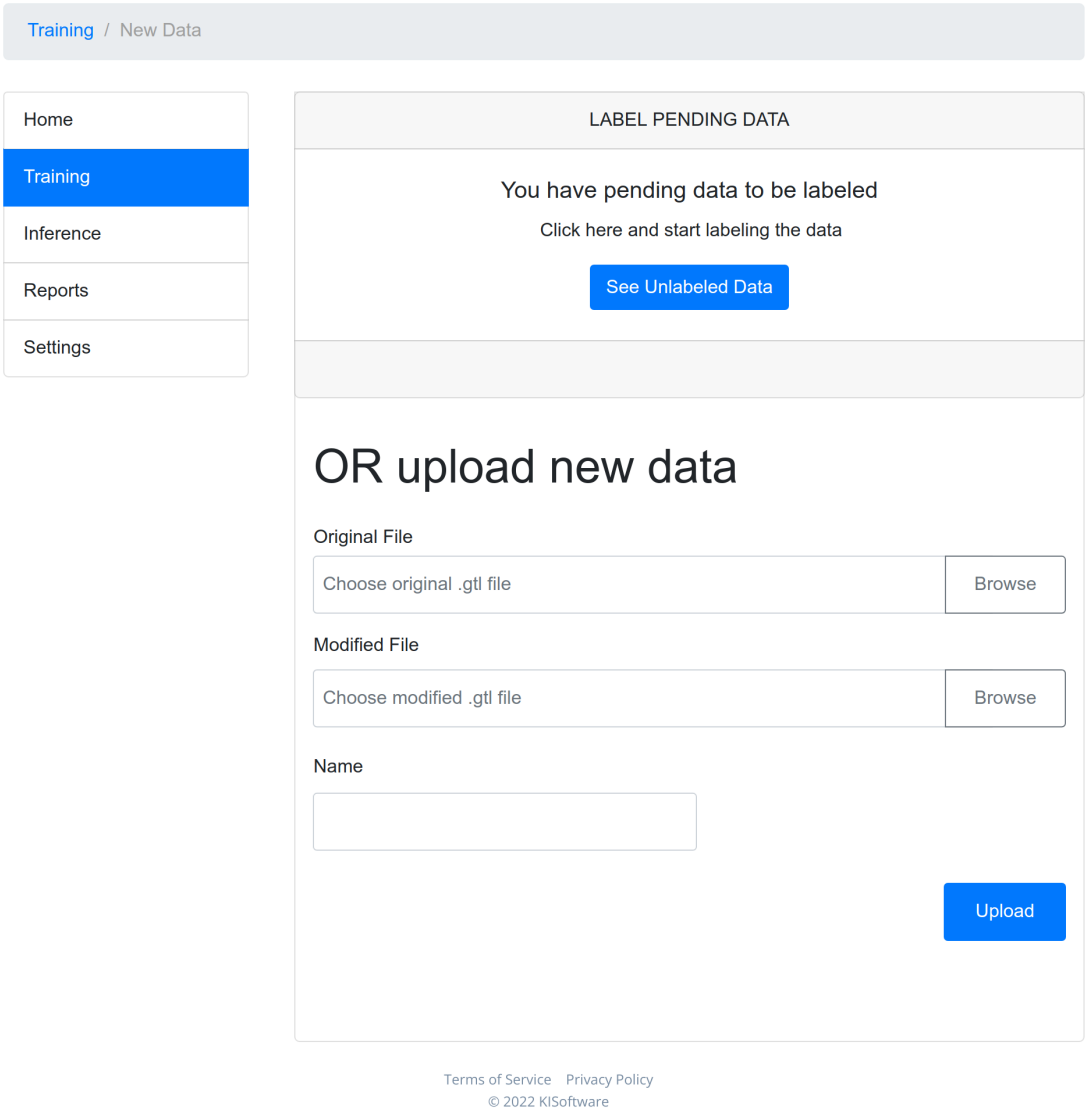

Figure 6: Wireframe of the new training flow page.

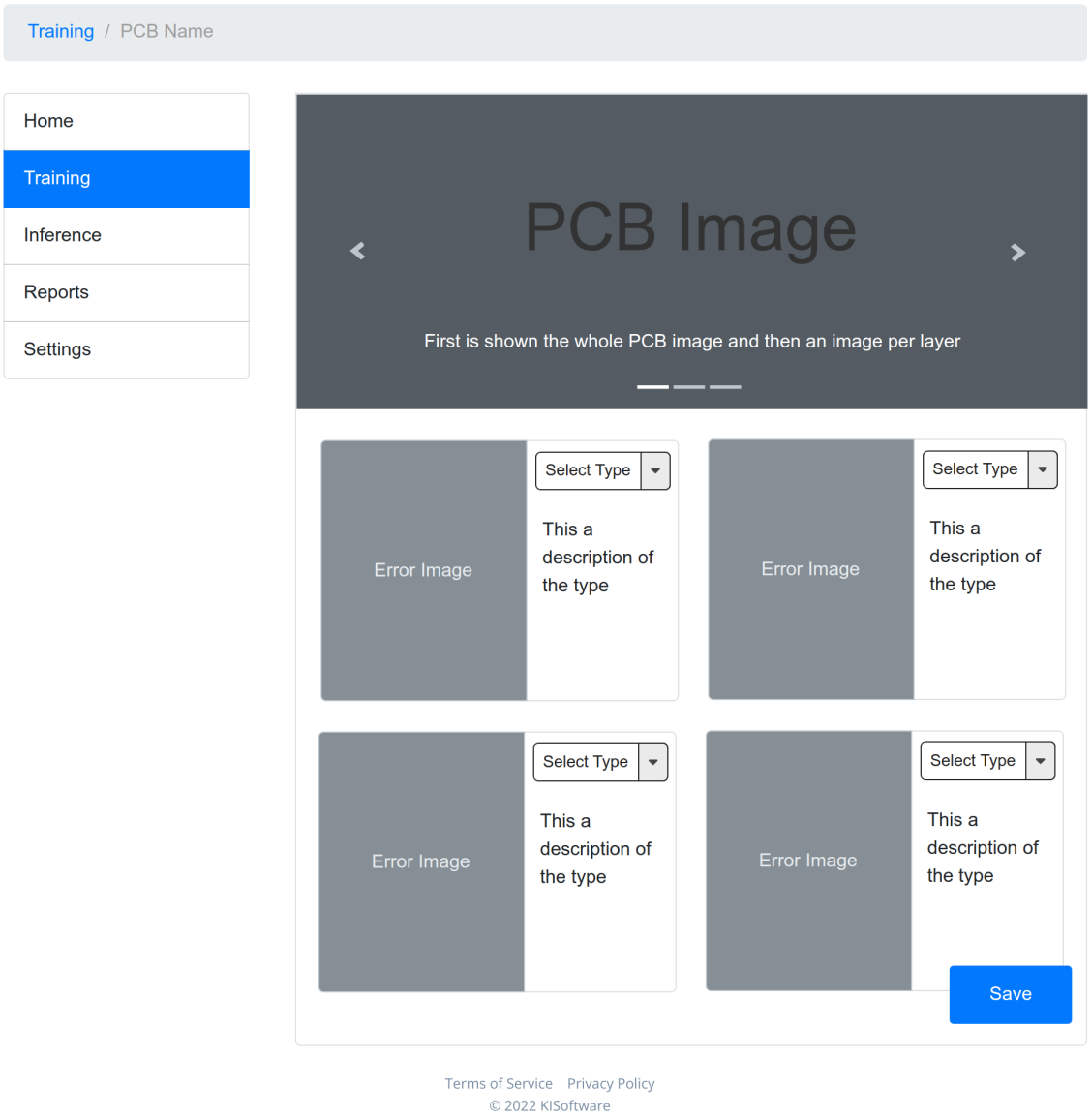

Figure 7: Wireframe of the training detail page.

Finally, we chose Yarn as the package manager for solving all the dependencies, libraries, and frameworks used by the frontend project.

Common concerns

Two repositories were created, one for the backend project and one for the frontend project in the IMMS Gitlab instance. We follow a branching strategy of a new branch for every new feature, and once the feature is developed, it is merged into the develop branch.

We used the Test Development Driven (TDD) in order to develop the backend project. This way we ensure a tested API for the most part and prevent potential bugs. The idea is to always create the test first and make it fail, and then develop the necessary code in order to make it pass with success. We also relied on the GitLab issues feature for managing the creation of tasks and bugs tickets on each project.

Final Result

As a result of applying the concepts, techniques, and tools mentioned in the previous chapters, we produced a tool that can support the defined Training and Inference process. This tool is a software that has two main components.

The first one is the backend, responsible for providing all the API endpoints for managing the training, inference, error type, and user flows and all the logic and data of the software. The backend can be deployed in any operating system that can run Docker, making it deployable in most popular operating systems without any advanced knowledge of any of the technologies used for developing it like Python, FastAPI, PostgreSQL, or any other. Just with a few Docker-related commands. It is important to emphasize that thanks to the TDD technique, it has more than 180 unit tests, which makes it have a test coverage of more than 90%. The second component is the frontend, a web desktop application capable of being installed in any operative system like Windows, Linux, or macOS with a user-friendly UI.

All the source code is hosted in the IMMS gitlab instance in two different repositories, each one with a readme file with all the necessary instructions to set up the solution from zero.

The following figures will show the current look of the main flows.

Authentication flow

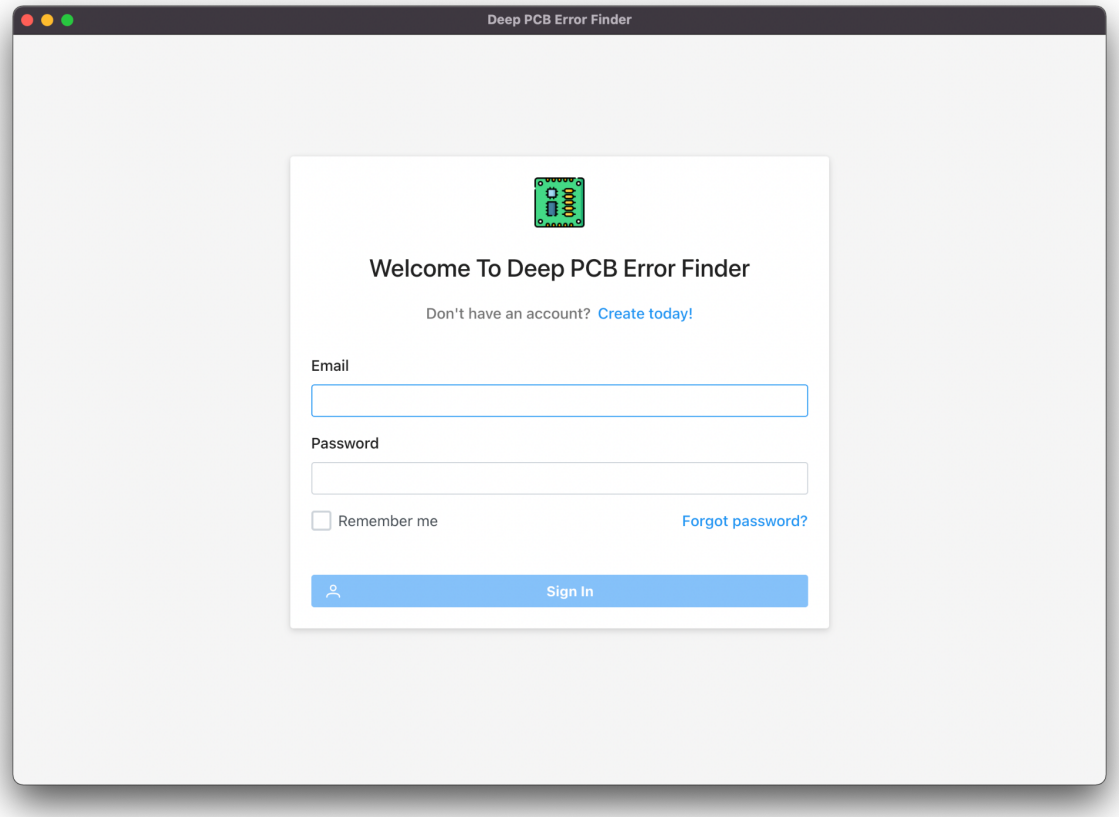

Figure 8: Login sceren.

Error type flow

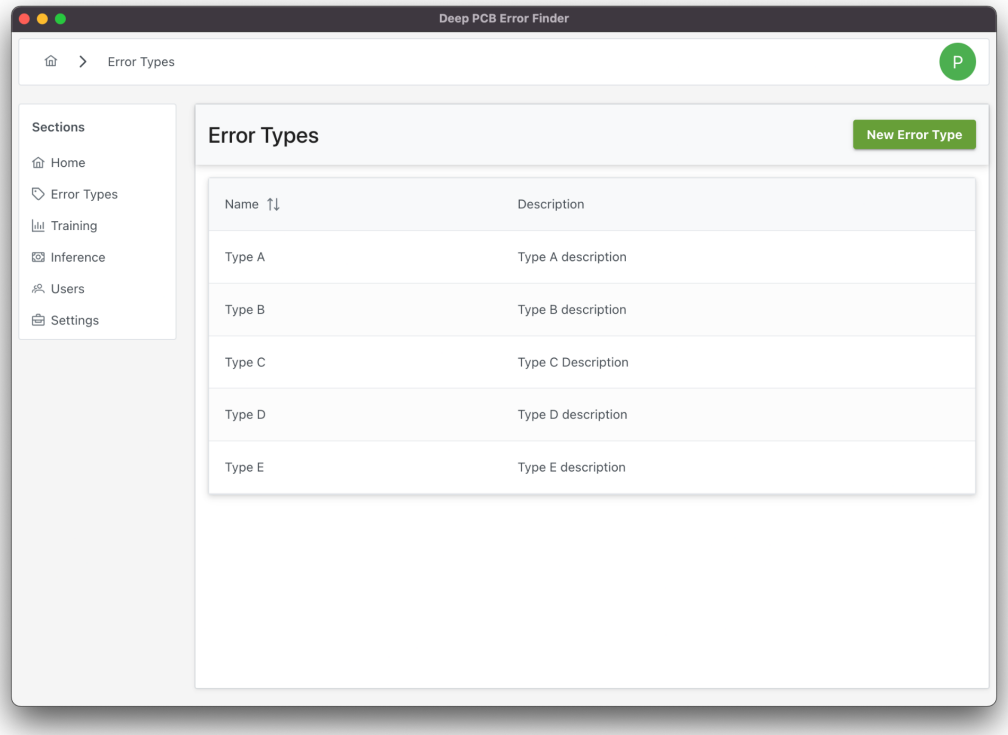

Figure 9: Error types list screen.

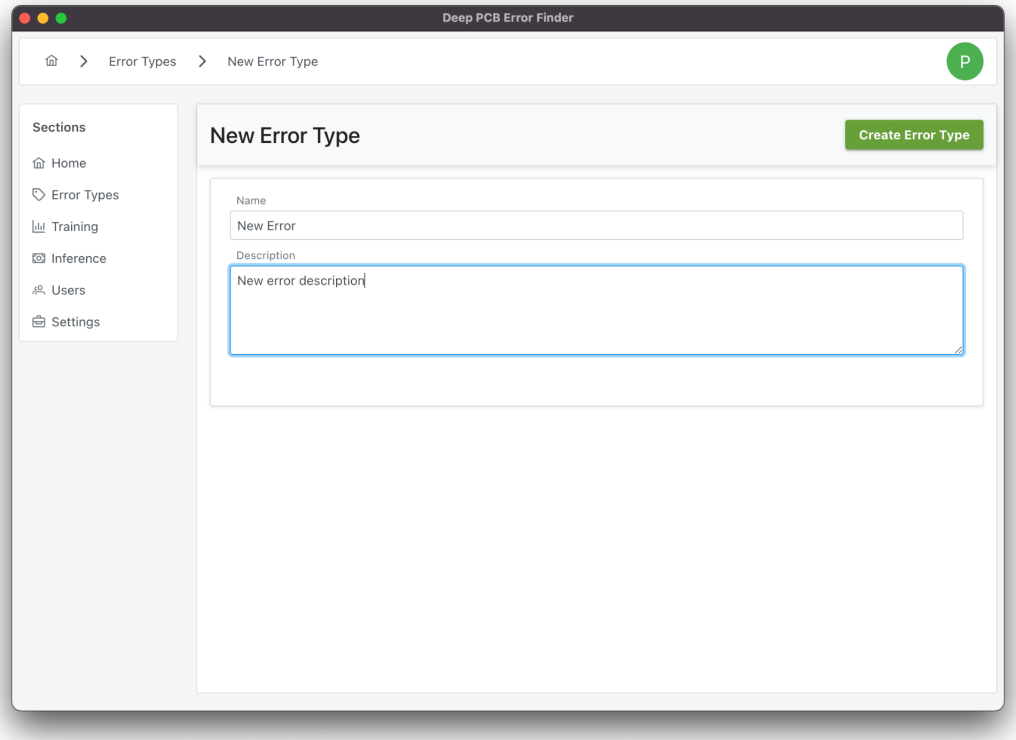

Figure 10: Error types list screen.

Training flow

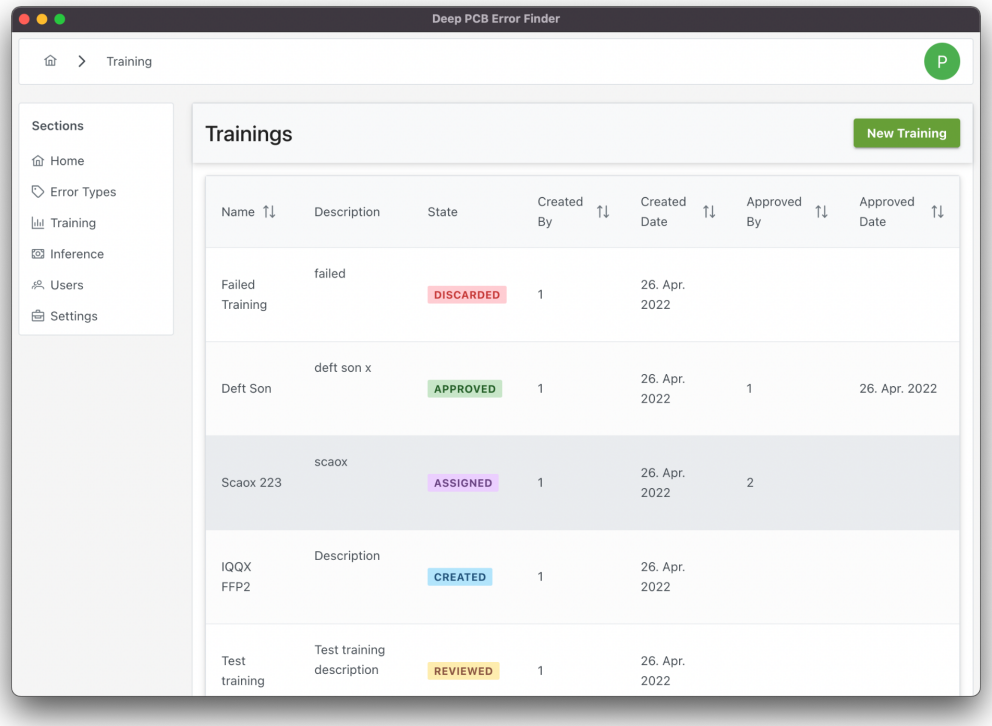

Figure 11: Training list screen.

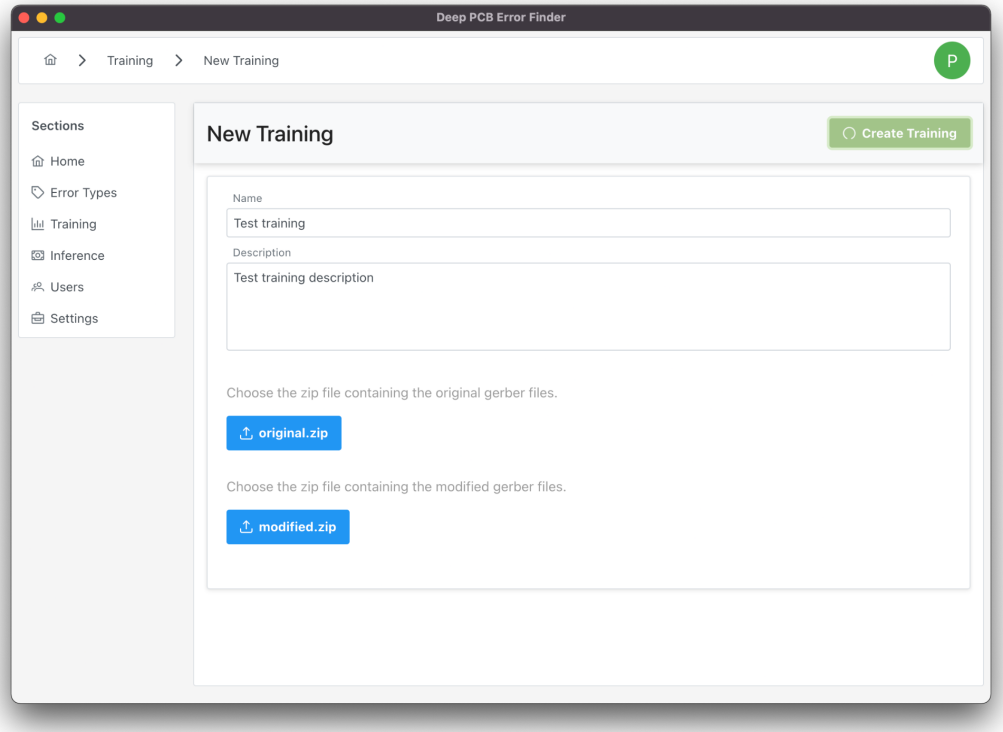

Figure 12: Training creation screen.

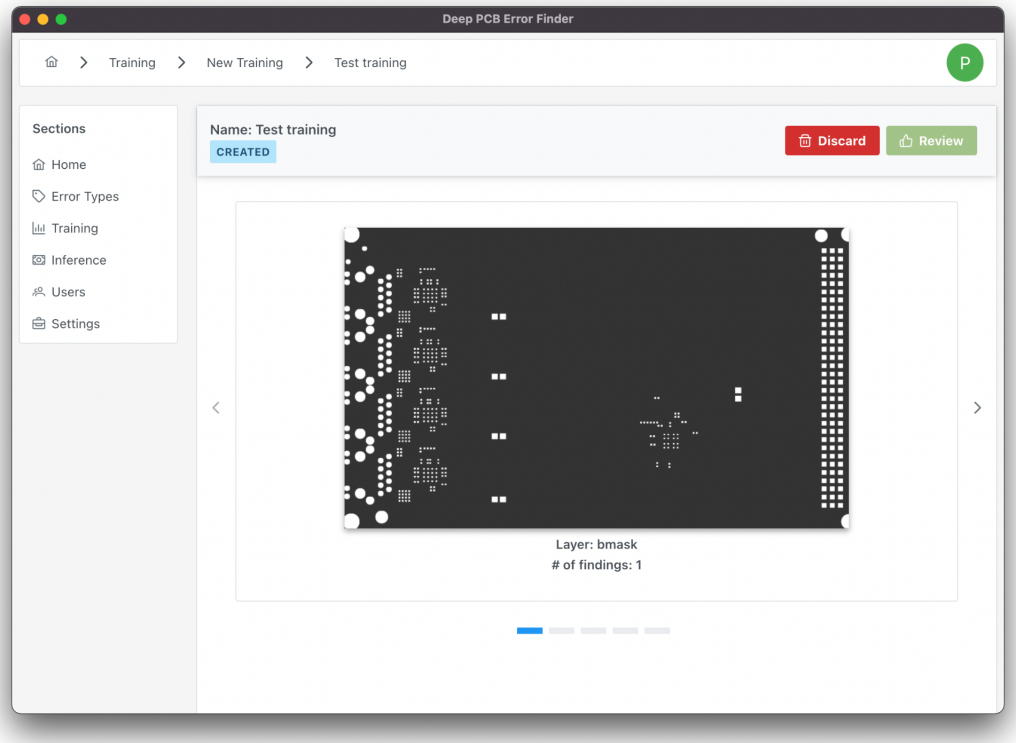

Figure 13: Training detail screen. The layer image is displayed.

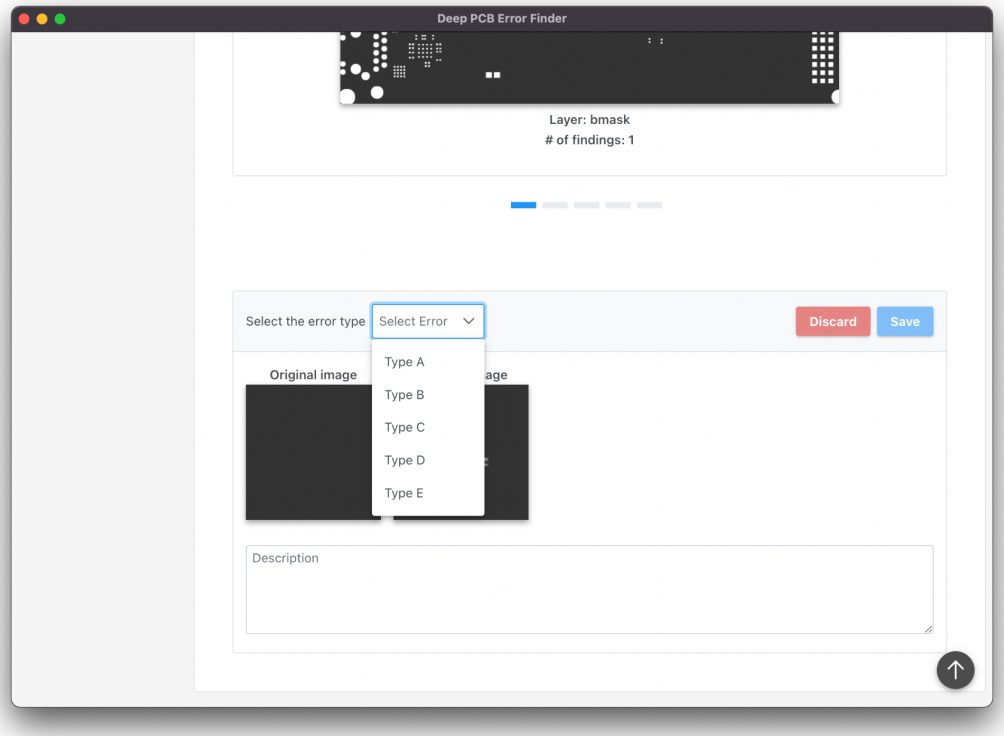

Figure 14: Training detail screen. A found difference tagging is being tagged.

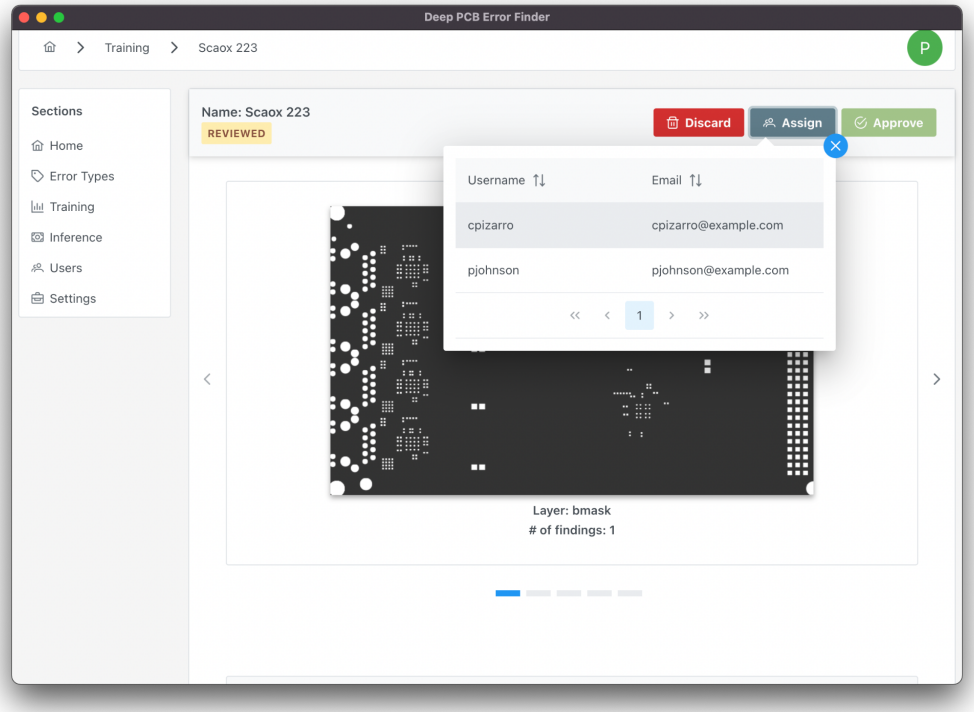

Figure 15: Training detail screen. A training is being assigned to another user for reviewing.

Inference flow

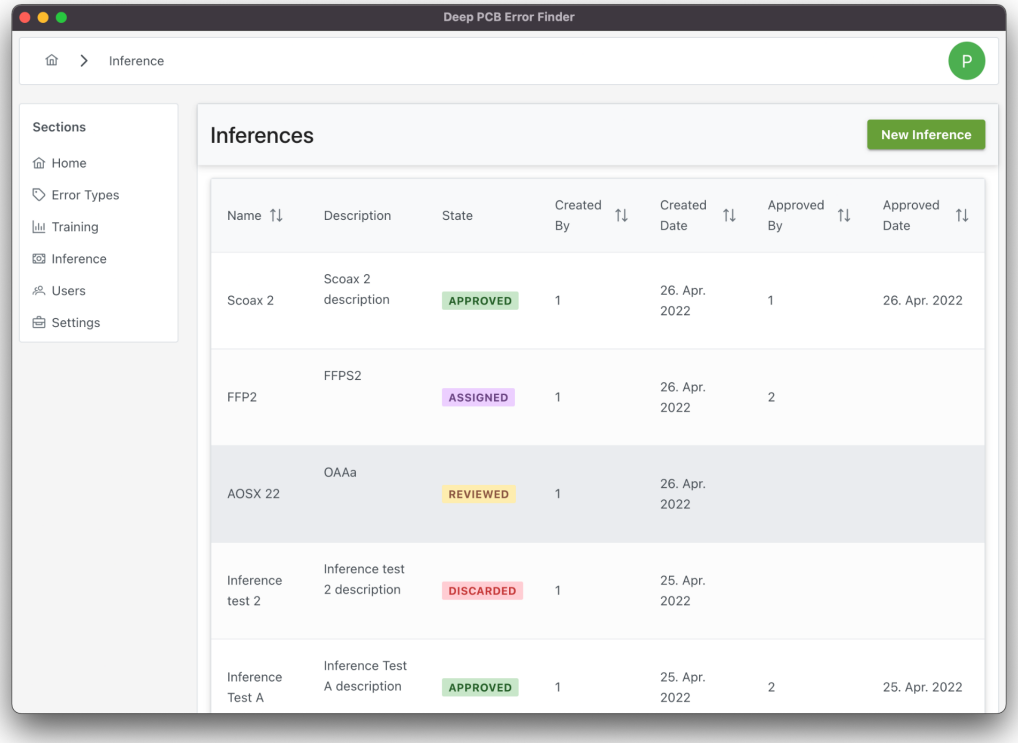

Figure 16: Inference list screen.

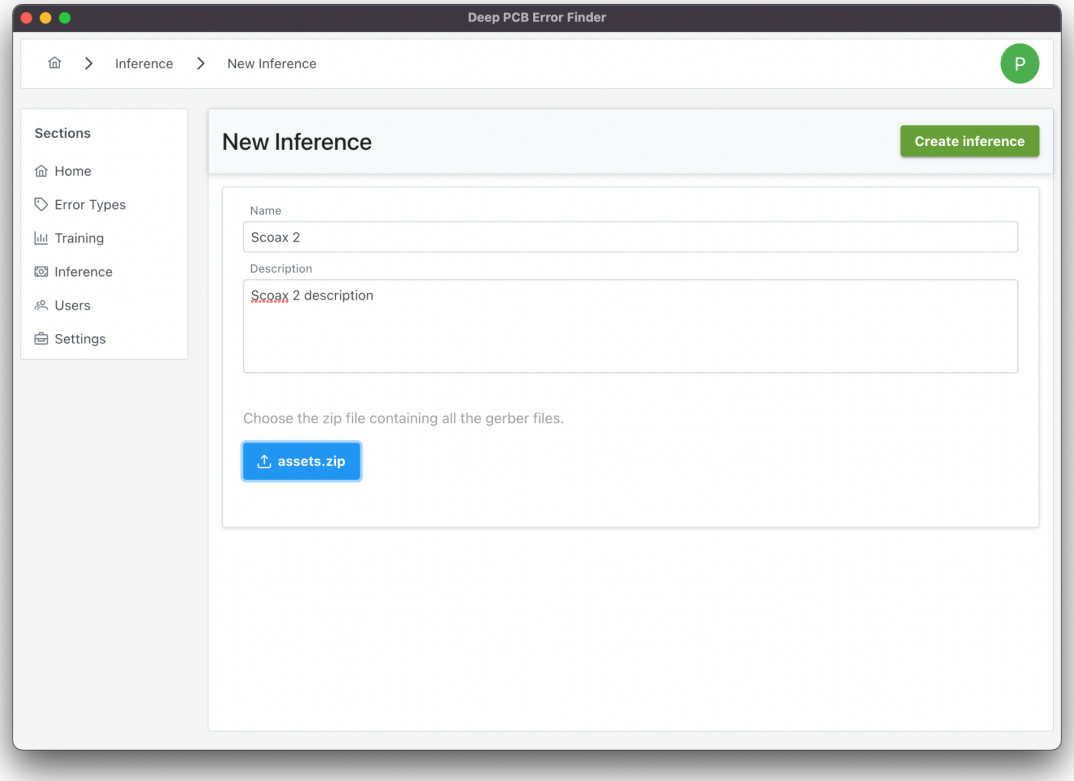

Figure 17: Inference creation screen.

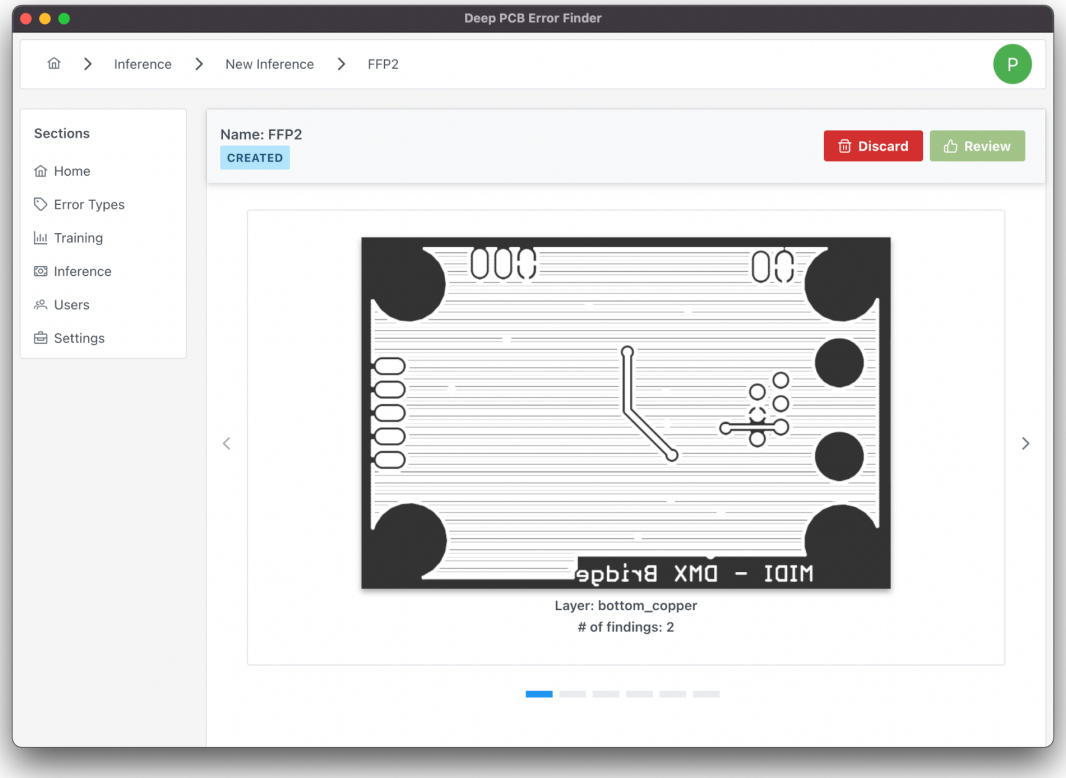

Figure 18: Inference detail screen.

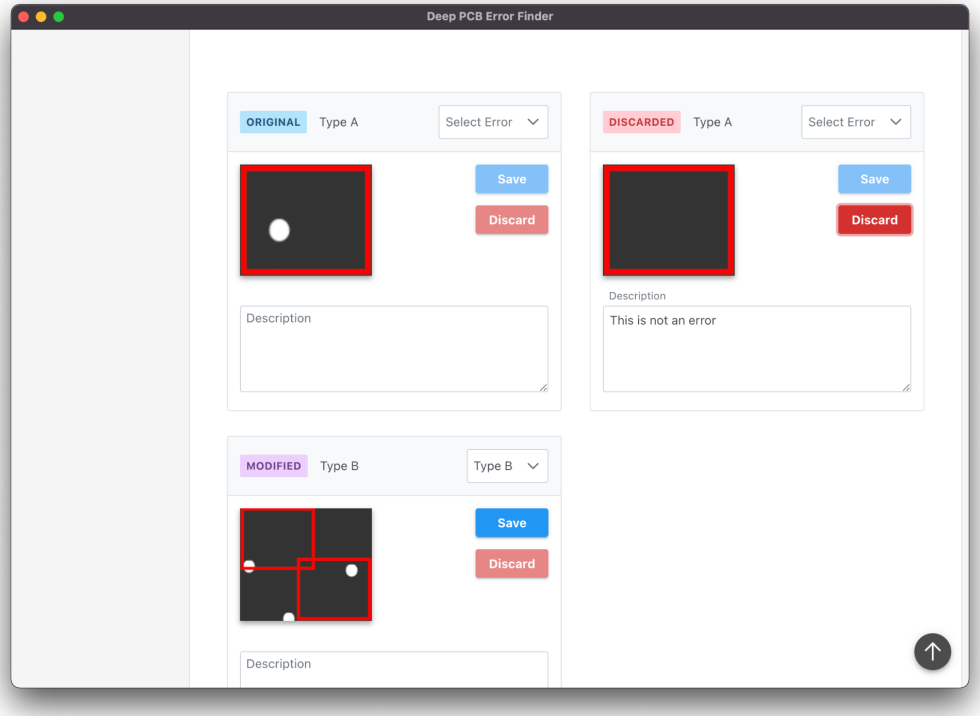

Figure 19: Inference detail screen. All the findings are shown. User is able to change the infered error type.

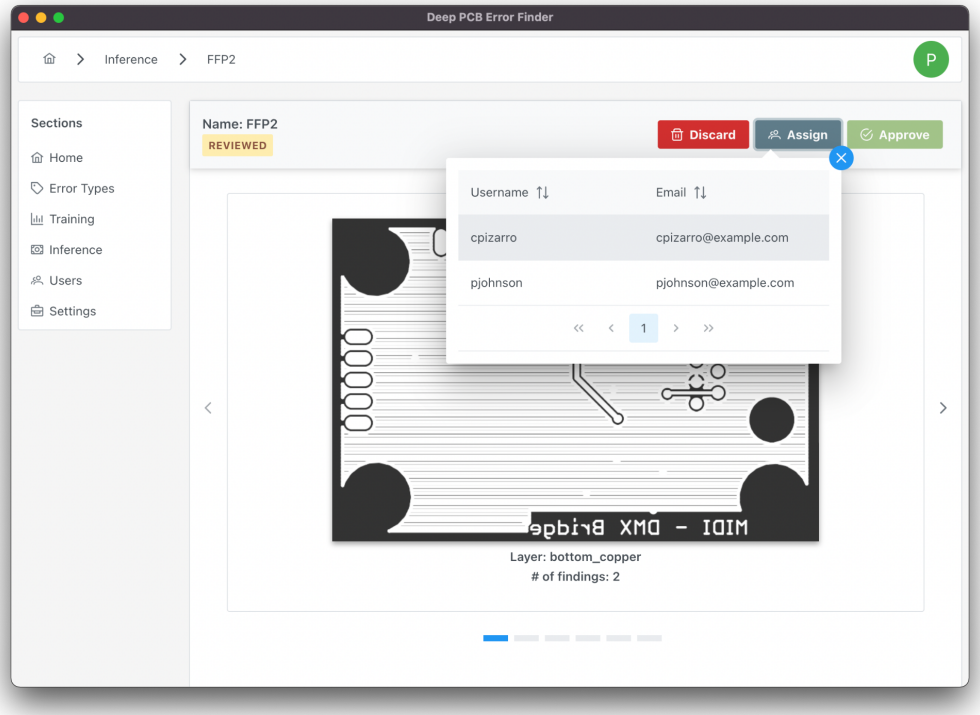

Figure 20: Inference detail screen. An inference is being assigned to another user for reviewing.

Users flow

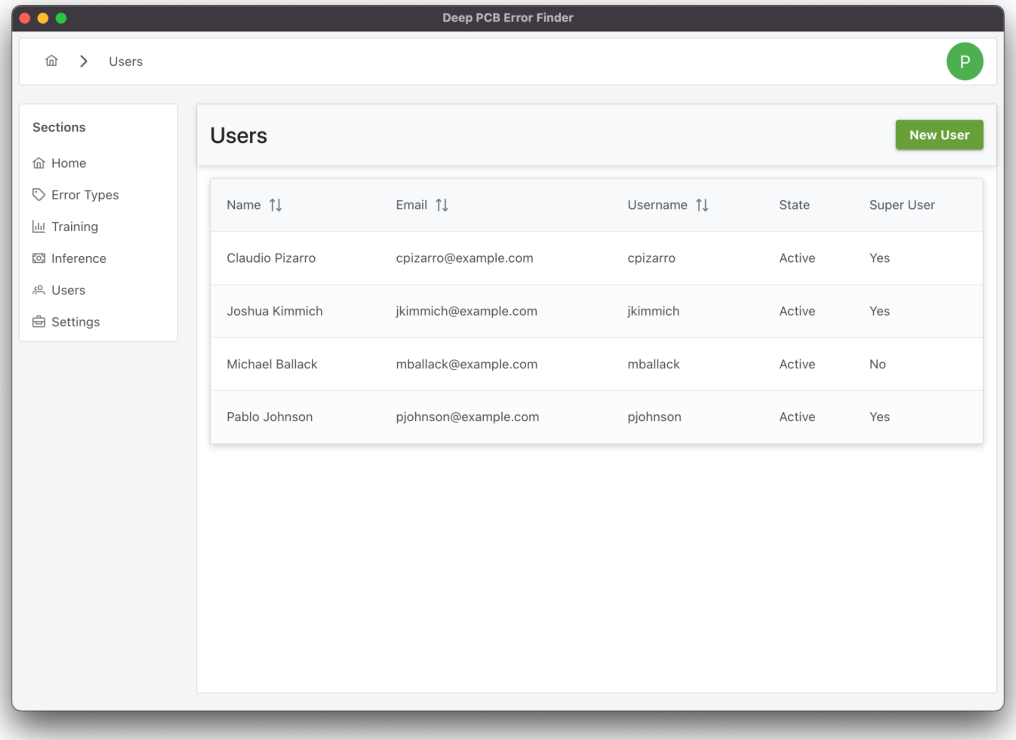

Figure 21: Users list screen.

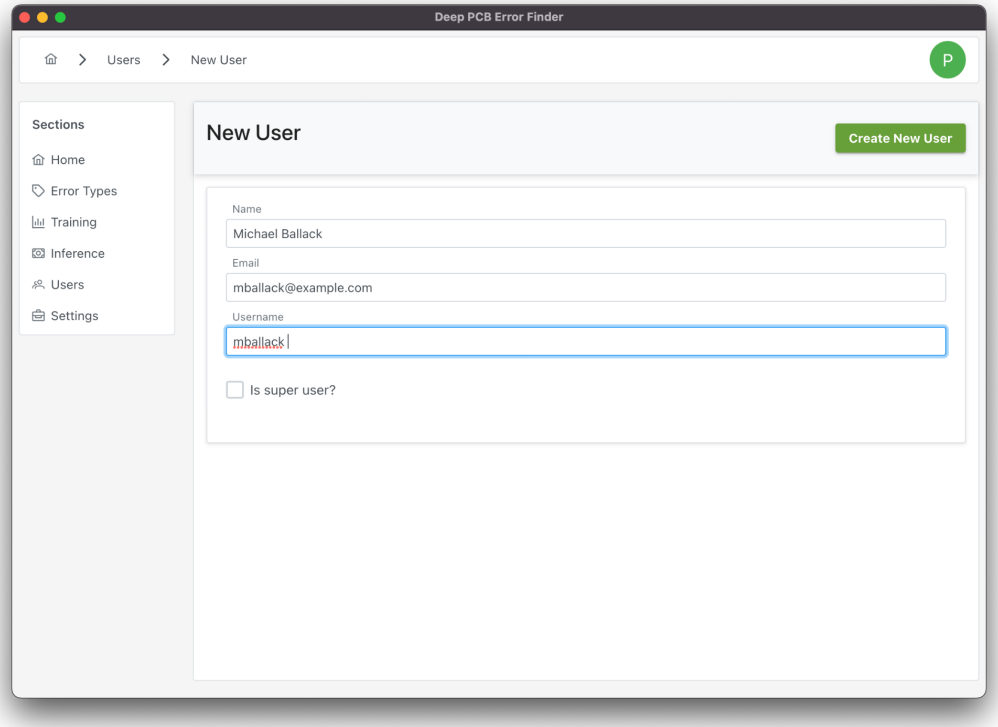

Figure 22: Users creation screen.

Personal Summary

This internship has been genuinely beneficial for me personally and in my career. It has allowed me to know from the inside how the environment in a purely German company is and to enjoy and meet new colleagues with whom, even though the internship was not very long, I was able to have some conversations about work and non-work related topics. Also, I was able to deepen my learning of German; even though most of the meetings I participated in were in English, there were times when we spoke in German, and I was able to exercise my knowledge of the language. From the professional point of view, I have been able to put into practice the experience gained in more than ten years of professional life and the knowledge gained so far in the master's program.

Essential courses such as deep learning allowed me, although, in this internship, no artificial intelligence or machine learning work was done, to understand what is needed to train and use a model, which was vital at the time of the analysis of the solution. Another course that served me as a basis for a good analysis and development of the solution was the Software Architecture course. This course was crucial to making decisions directly related to the code and the way to solve some problems.

In addition, the practice allowed me to deepen my knowledge of Python, which I consider fundamental for the career path I want to follow, as well as experiment with new technologies/tools such as FastAPI, Vue, and Docker.

Finally, I am pleased and satisfied to have done my internship at IMMS. I feel that it has been a place that has allowed me to learn and share my knowledge freely.