



Hochschule für Technik  
und Wirtschaft Berlin

University of Applied Sciences

**Project Studies**  
**[Semester 2: Pro-ITD]**

**Project Report**  
**on**  
**“PUT IT IN THE BOX”**

**By:** Anusree Preetha Ranjith (s0585834) & Sharanya Adiga (s0585849)

**To:** Prof. Dr.-Ing. Thomas Schwotzer and Prof. Dr.-Ing. Katarina Adam

**Date:** 11.07.2023

## **Table of contents**

Acknowledgment	3
List of tables	4
List Figures	5
<b>1. Abstract</b>	<b>6</b>
<b>2. Introduction</b>	<b>7</b>
<b>3. Outline of the problem</b>	<b>8</b>
<b>4. Methodology and approach</b>	<b>9</b>
4.1 System functional requirements	9
4.2 System non functional requirements	9
<b>5. Analysis</b>	<b>10</b>
5.1 User story	10
5.2 Scope for project	11
<b>6. Implementation</b>	<b>12</b>
6.1 Setup	12
6.2 UML sequence diagram	16
<b>7. Approach</b>	<b>18</b>
7.1 Login and Structure of the PDF	18
7.2 Uploading	18
7.3 Extraction	19
7.4 Notifying	20
7.5 Encryption	21
<b>8. Testing</b>	<b>23</b>
<b>9. Summary</b>	<b>26</b>
<b>10. Conclusion and future work</b>	<b>27</b>
<b>Reference</b>	<b>28</b>

# Acknowledgment

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of the people who made it possible whose constant guidance and encouragement crowned the efforts with success.

Without the constant support of our supervisors, **Prof. Dr.-Ing. Thomas Schwotzer** and **Prof. Dr.-Ing. Katarina Adam** this project and the research that went into it would not have been feasible. From our initial interaction regarding the project to the final draft of this dissertation, their zeal, knowledge, and meticulous attention to detail have been an inspiration and kept our work on track.

This report corresponds with the Project Studies module of the Degree Programme of the “Professional IT Business and Digitalization” and therefore is intended to shed light on the various phases we went through in order to achieve the desired results.

Anusree Preetha Ranjith  
Sharanya Adiga

## List of tables

1. User story	10
2. Manual Testing Result	25

## List of figures

1. Visual studio code installation	12
2. Node.js installation	14
3. EmailJS template	15
4. UML Sequence diagram	17
5. Login page	18
6. Uploading a .pdf file	19
7. Extracting the details	20
8. Notifying	21
9. E2E Testing	23

## 1. Abstract

The housing allowance calculation, or **WOHNGELDABRECHNUNG** in German, refers to the calculation and allocation of housing benefits. Wohngeld is a state subsidy available to low-income households to help cover their housing costs. The specific process of Wohngeldabrechnung may vary depending on the country and legal regulations. In general, the housing allowance calculation is based on various factors such as the total income of the household, the number of family members, the cost of rent or mortgage payments, and any additional applicable expenses. The calculation takes into account the specific eligibility criteria and formulas set by the respective government authority responsible for administering housing benefits.

Once all the necessary information has been gathered, the Wohngeldabrechnung determines the amount of housing allowance that the household is eligible to receive. This calculation is typically performed periodically, such as monthly or annually, to ensure that the housing benefits accurately reflect the current circumstances of the household. This paper presents an approach to extract details from Wohngeldabrechnung using React[1], a popular JavaScript[2] library for building user interfaces.

## 2. Introduction

WOHNGELDABRECHNUNG[1] or housing allowance calculation, is that it provides financial support to low-income households for covering their housing costs. This assistance can be crucial in ensuring that individuals and families have access to safe and affordable housing.

**PUT IT IN THE BOX** is a user-friendly website designed to extract the necessary information from your Wohngeldabrechnung documents. Whether you're an individual seeking housing benefits or a housing authority professional responsible for processing applications, our platform is tailored to meet your needs. With our advanced technology and intuitive interface, extracting details from Wohngeldabrechnung has never been easier. Simply upload your document, and our intelligent system will swiftly analyze and extract the relevant information, such as bank, tax details of the user.

A more efficient system can be built using modern web technologies such as React and javascript, which allows for the creation of interactive and dynamic user interfaces. This paper describes the key features of a course request system built using React, including extracting bank and tax details, displaying them along with notifying the respective officers regarding the same.

The objective of this document is to shed light on the significance of housing allowance calculation, provide insights into the Wohngeldabrechnung process, and present a potential solution for extracting relevant details. It is our hope that this exploration contributes to the understanding and improvement of housing benefit systems, ultimately benefiting low-income households and ensuring access to adequate housing.

1. <https://exporo.de/wiki/wohngeldabrechnung/#:~:text=In%20der%20Wohngeldabrechnung%20sind%20s%C3%A4mtliche,Jahr%20von%20der%20Verwaltung%20erstellt.>

### 3. Outline of the problem

#### **To outline the obstacles we faced while building the website**

The first step is the Installation process. We installed Visual Studio code and React to the Systems as we had different OS along with some initial npm errors.

The next step was Data extraction and analysis where we focused on developing an intelligent system that can accurately extract relevant information, such as bank and tax details, from uploaded Wohngeldabrechnung documents. It requires implementing advanced technologies, such as optical character recognition (OCR) and natural language processing (NLP), to parse and analyze the documents effectively.

Next we focused on User interface and experience for designing a user-friendly interface that is intuitive and accessible to individuals of varying technological proficiency is essential. The website should be easy to navigate, with clear instructions and visual cues to guide users through the document upload process and provide feedback on the progress.

The fourth step was to improve the scalability and performance of the web application since the number of users and uploaded documents increases, the website should be able to handle the growing volume of data and maintain optimal performance.

Finally , we deal with Compliance with regulations: Building the website in compliance with relevant regulations, such as data protection laws, accessibility guidelines, and housing benefit program requirements, is crucial. Staying up to date with any regulatory changes and incorporating necessary adjustments to the website is essential for long-term viability.

1. <https://tools.pdf24.org/en/ocr-pdf>

2. The Giles Ecosystem – Storage, Text Extraction, and OCR of Documents



## 4.Methodology

The approach methodology and environment for Put it in the Box are explained in this section. The web-based system was developed using the Agile Software development approach in which the system is divided into different parts and executed sequentially in order to achieve distinct highlighted goals. This section focuses on the functional and nonfunctional requirements of the system, use-case diagrams and specifications of each user of the system, and finally the class diagram showing the interconnection between system objects.

### 4.1 System Functional Requirements

Functional requirements specify the system's functions and services. The following are the Functional Requirements:

1. Designated users should be able to login with their credentials and be routed to
2. appropriate screens based on their access levels.
3. The User can upload the document.
4. The respective data can be extracted from the document.
5. The data includes tax and bank details of the user.
6. The Bank officers are notified with the extracted bank information via mail.
7. Similarly The tax officers are notified with the extracted tax information via mail.

### 4.2 System Non-Functional Requirements

Non-functional requirements define how the system functions to the system limitations. It demonstrates the system quality. The non-functional requirements of the system are as follows.

The system should be:

1. Simple and user-friendly
2. Efficient and accurate
3. Secure access to authorized users
4. Testable, Reliable and Scalable
5. Cost-effective
6. Robust so as to handle errors
7. Secure privacy of users' information.

## 5. Analysis

### 5.1 User story

In software development and product management, a user story is an informal, natural language description of one or more features of a software system. A user story[1] is a tool used in Agile software development to capture a description of a software feature from an end-user perspective. A user story describes the type of user, what they want and why. A user story helps to create a simplified description of a requirement. A user story is an informal, general explanation of a software feature written from the perspective of the end user. Its purpose is to articulate how a software feature will provide value to the customer.

ID	Name	Story	Acceptance criteria
01	Sign Up	The User will be able to sign up on the site.	Users can create a new account by using Username.
02	Sign In	The User will be able to login to the site.	Users can login to the site using Username and password.
03	Upload a file	The User can upload a file on the website.	Users are allowed to upload .pdf files.
04	Extract Bank details	The User can extract Bank details from the site.	Users can extract bank details like Name, IBAN and BIC.
05	Extract Tax details	The User can extract Tax details from the site.	Users can extract tax details like Tax ID, type and category.
06	Notify Bank Officer	The User can notify Bank officers on the site.	Users can notify bank officers with the extracted bank details.
07	Notify Tax officer	The User can notify Tax officers on the site.	Users can notify tax officers with the extracted tax details.

**Table 1: User story**

1. <https://www.figma.com/file/kd9tKiALlAC2mOvI1H10a2/PROJECT?type=design&node-id=0-1&mode=design&t=sxWtsISyUFZ8EAxV-0>

## 5.2 Scope for project

The scope of the project focuses on developing a solution that utilizes React, a JavaScript library for user interface development, to extract details from WOHNELDABRECHNUNG[1] (housing allowance calculation). The project aims to address the following key aspects.

Firstly, a comprehensive understanding of the **housing allowance calculation** process is necessary. This involves gaining insights into the factors considered, eligibility criteria, and the periodic review process. By understanding these aspects, the project team can develop a solution that accurately extracts the relevant details from Wohngeldabrechnung documents.

The **Data Extraction**[2] [3] approach will leverage React to efficiently process and extract the necessary information from Wohngeldabrechnung documents. This approach will involve exploring various techniques for parsing and processing data, ensuring accuracy and reliability in the extraction process. By utilizing React, the team can leverage its robust ecosystem of libraries and tools to streamline the data extraction process effectively.

To provide a user-friendly experience, the project will involve developing a **User Interface** using React. The interface will facilitate the extraction of information from Wohngeldabrechnung documents. The design of the interface will prioritize intuitiveness and ease of use, allowing users to input relevant data and retrieve calculated housing allowances effortlessly. By leveraging React's component-based architecture and UI capabilities, the team can create an interface that meets the usability requirements of the project.

To ensure the functionality, performance, and reliability of the solution, thorough **Testing** will be conducted. The testing phase will validate the solution's ability to accurately extract and process data from Wohngeldabrechnung documents. Additionally, quality assurance measures will be implemented to ensure that the solution meets the required standards and specifications. This includes conducting rigorous testing scenarios, addressing edge cases, and implementing error handling mechanisms.

**Comprehensive documentation** will be prepared to outline the architecture, functionality, and usage guidelines of the solution. This documentation will serve as a reference for stakeholders involved in the project, including end-users, administrators, and maintenance personnel. Additionally, knowledge transfer activities will be conducted to ensure that relevant stakeholders have a clear understanding of the solution's implementation and can effectively utilize and maintain it in the future.

1. <https://finanzierung.com/glossar/wohngeldabrechnung/>
2. A new method of information extraction from PDF files.
3. Identification and extraction of different objects and its location from a Pdf file using efficient information retrieval tools

## 6. Implementation

### 6.1 Set up

#### 1. Visual studio code installation:

To install **Visual Studio Code (VS Code)**[1], following are the steps:

1. Visit the official website of Visual Studio Code at <https://code.visualstudio.com/>.
2. Downloading depends on the operating system (Windows, macOS, or Linux).
3. Once the download is complete, locate the installer file and run it.
4. Follow the on-screen instructions provided by the installer.
5. On Windows, you may need to confirm User Account Control prompts or grant necessary permissions during the installation process.
6. After the installation is complete, launch Visual Studio Code.
7. Upon launching, you will see the welcome screen of Visual Studio Code.

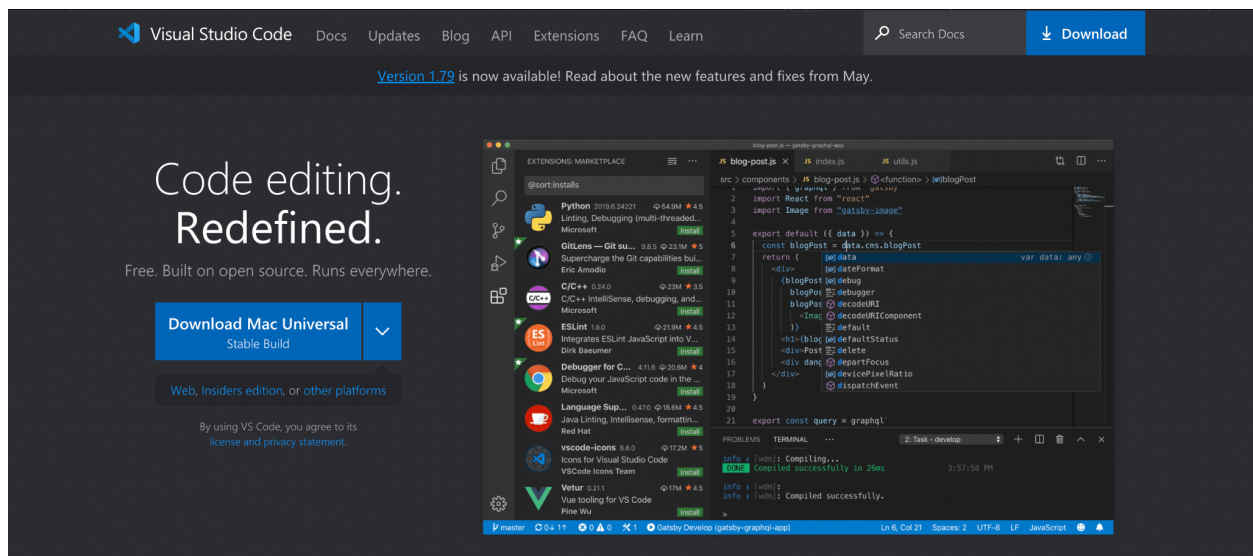


Figure 1: Visual studio code installation.

#### 2. React installation and Javascript:

##### Set up Node.js and npm

1. <https://code.visualstudio.com/>

React requires Node.js and npm [1] to be installed on your computer. Then we visit the official Node.js website (<https://nodejs.org>) and download the recommended LTS (Long-Term Support) version for the operating system. Run the installer and follow the instructions to install Node.js.

Once installed, open command-line interface (such as Terminal or Command Prompt) and verify that Node.js and npm are installed by running the following commands:

```
node -v
npm -v
```

### **Create a React project**

To set up a new React project [2], you can use the "Create React App" command-line tool provided by React. First, ensure you have Node.js and npm installed on your system. Then, open your command-line interface and run `npm install -g create-react-app` followed by `npx create-react-app my-react-app` once the installation is complete. Navigate into the project directory and start the development server by running:

```
cd my-react-app
npm start
```

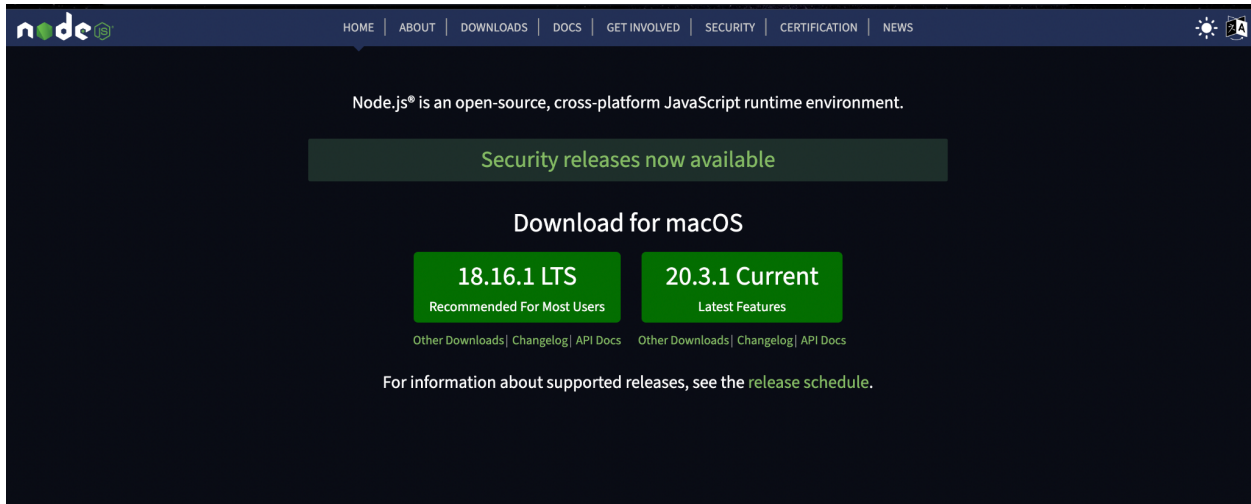
This will launch the development server, and you can view your React application by opening a web browser and accessing <http://localhost:3000>.

### **Start the development server**

To navigate to the project directory, run the following command: `cd my-react-app`. After executing this command, we will be in the directory where our React application is located. To start the development server, use the command `npm start`. Running this command will launch the development server, and our React application will open in our default web browser. The development server has a hot-reload feature, which means that any changes we make to our React code will be automatically reflected in the browser. This allows us to see the updates in real-time without having to manually refresh the page.

By following these steps, we can easily set up our development environment and begin working on our React application.

1. <https://nodejs.dev/en/>
2. <https://react.dev/learn/start-a-new-react-project>



**Figure 2: Node.js installation**

We chose Javascript because,

All major web browsers support JavaScript, making it a widely accessible language for web development. JavaScript engines optimize the code before executing it, resulting in excellent performance even if the code is not highly efficient. This aspect allows developers to focus more on writing readable and maintainable code rather than obsessing over performance concerns. In comparison to languages like Python or Ruby, JavaScript's execution speed is particularly notable.

JavaScript boasts an extensive collection of libraries and packages available through npm (Node Package Manager). The npm repository hosts a vast number of libraries, providing developers with a wide range of tools and functionalities to enhance their projects. Moreover, JavaScript's package management system, package.json, offers a straightforward and efficient way to handle project dependencies, simplifying the development process.

One of JavaScript's strengths lies in its ability to provide various interfaces for building engaging and interactive websites. This versatility enables developers to create dynamic and responsive web applications, leading to enhanced user interaction and overall user experience. By utilizing JavaScript, developers can employ a diverse set of techniques and technologies to captivate users and make their websites more engaging.

### **3.EmailJS:**

**To use EmailJS [1] for sending emails from the React application, below steps are followed:**

1. <https://www.emailjs.com/>

To get started with EmailJS, you can visit their website at <https://www.emailjs.com/> and sign up for an account. Once you have signed up, you will gain access to your EmailJS dashboard.

In your EmailJS dashboard, navigate to the "Email Templates" [1] section. From there, you can create a new template by providing a name, subject, and content for your email. Customize the template according to your specific needs, including any necessary placeholders for dynamic data.

**Figure 3: EmailJS template**

Next, open your command-line interface within your React project directory. Run the following command to install the EmailJS library: "npm install emailjs-com". This will add the necessary dependencies to your project.

In the React component file where you want to send an email, you need to import the EmailJS library. You can do this by adding the following line at the top of your component file "import emailjs from 'emailjs-com';"

Configure your EmailJS settings by providing your User ID and the Template ID you created in your EmailJS dashboard and to send an email, use the sendTaxEmail function:

```
emailjs.init('your_user_id_here');
emailjs.send('your_service_id_here', 'your_template_id_here', {})
```

```
function sendTaxEmail() {

    const name = taxInfo.Name;
```

1. <https://dashboard.emailjs.com/admin/templates>

```
const TaxId = taxInfo.TaxId;
const TaxCategory = taxInfo.TaxCategory;
const SocialSecurity = taxInfo.SocialSecurity;
const TaxType = taxInfo.TaxType;
const amount = taxInfo.Amount;

const templateParams = {
  name, TaxId, TaxCategory, SocialSecurity, TaxType, amount
};

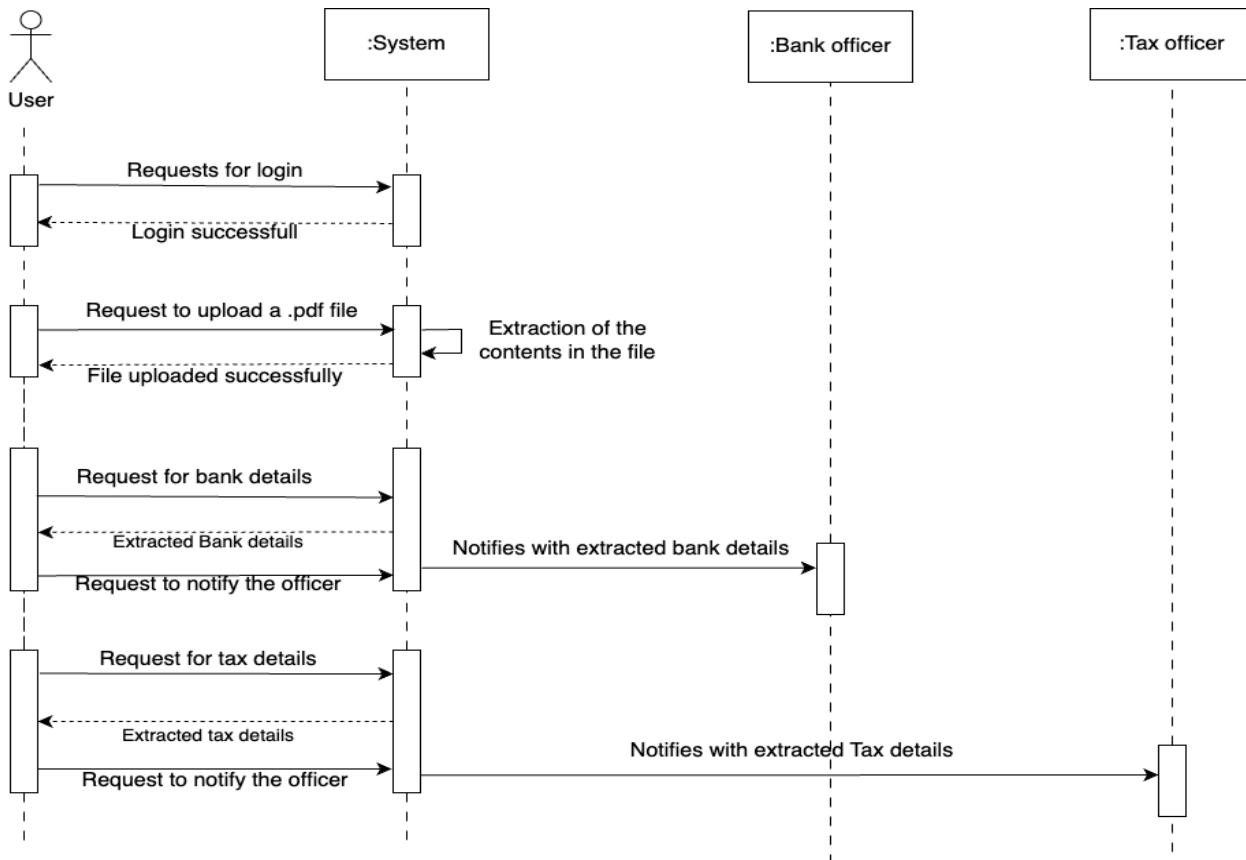
var recipient = 'example@example.com';
var subject = 'Hello';
var body = 'This is the email content.';
var data = {
  service_id: 'service_me3uxoz',
  template_id: 'template_ygfgdvy',
  user_id: 'PnSkZC0BI309uKdri',
  template_params: {
    recipient: recipient,
    subject: subject,
    body: body,
    ...templateParams
  }
};

axios.post('https://api.emailjs.com/api/v1.0/email/send', data)
  .then(function (response) {
    console.log('Email sent successfully:', response.data);
  })
  .catch(function (error) {
    console.log('Error occurred while sending email:', error);
  });
}
```

## 6.2 UML Sequence diagram



A sequence diagram is a Unified Modeling Language (UML) [1] diagram that illustrates the sequence of messages between objects in an interaction. A sequence diagram consists of a group of objects that are represented by lifelines, and the messages that they exchange over time during the interaction.



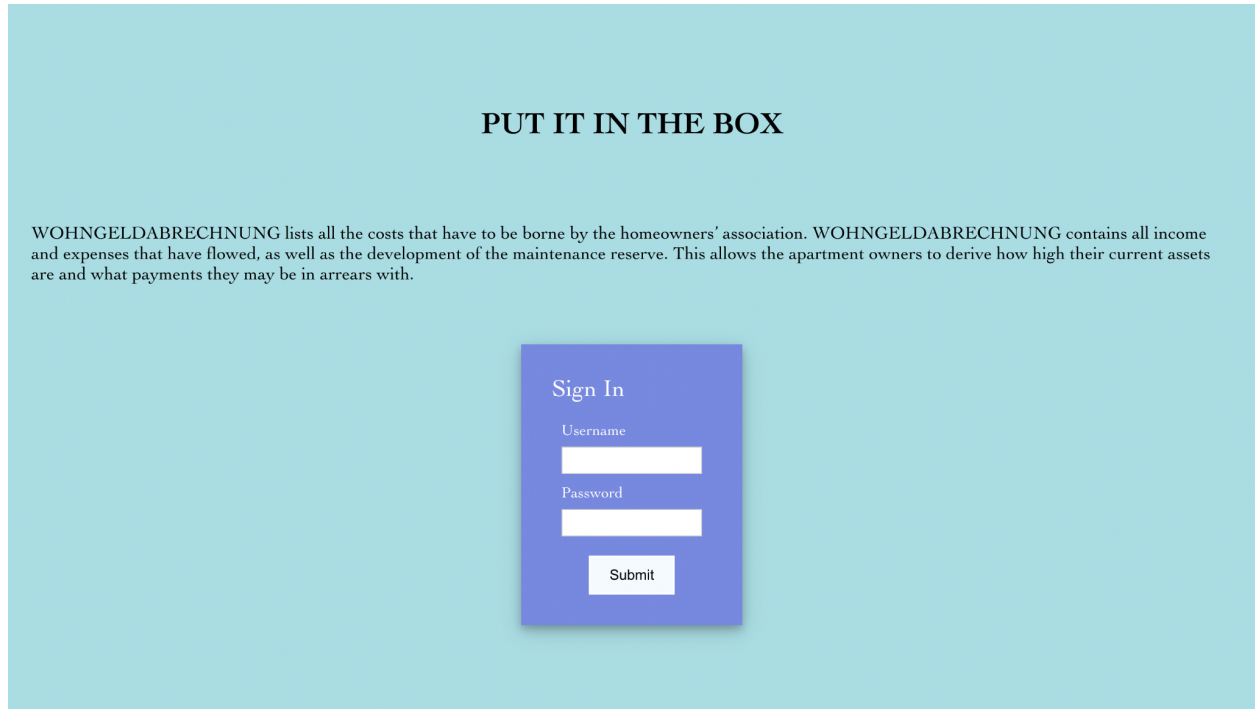
**Figure 4: UML Sequence diagram**

The user can upload the .pdf file and extract the respective details by requesting the system. Adding to this the user can also notify the respective officers with the extracted details.

1. [https://en.wikipedia.org/wiki/Unified\\_Modeling\\_Language#:~:text=The%20unified%20modeling%20language%20\(UML,the%20design%20of%20a%20system.](https://en.wikipedia.org/wiki/Unified_Modeling_Language#:~:text=The%20unified%20modeling%20language%20(UML,the%20design%20of%20a%20system.)

## 7. Approach

### 7.1 Login and Structure

The image shows a login page with a light blue background. At the top center, the text "PUT IT IN THE BOX" is displayed in a bold, black, sans-serif font. Below this, a paragraph of text explains that "WOHNGELDABRECHNUNG" lists costs for homeowners' associations and contains income and expense data for maintenance reserves. In the center of the page is a purple rectangular box with rounded corners and a subtle drop shadow. Inside this box, the text "Sign In" is at the top. Below it are two white input fields: the first is labeled "Username" and the second is labeled "Password". At the bottom of the purple box is a white button with the text "Submit".

**PUT IT IN THE BOX**

WOHNGELDABRECHNUNG lists all the costs that have to be borne by the homeowners' association. WOHNGELDABRECHNUNG contains all income and expenses that have flowed, as well as the development of the maintenance reserve. This allows the apartment owners to derive how high their current assets are and what payments they may be in arrears with.

**Sign In**

Username

Password

Submit

**Figure 5: Login page**

The structure of the pdf includes the following :

**user address , contract id , accounting period**

**tax id , tax category, tax type , amount**

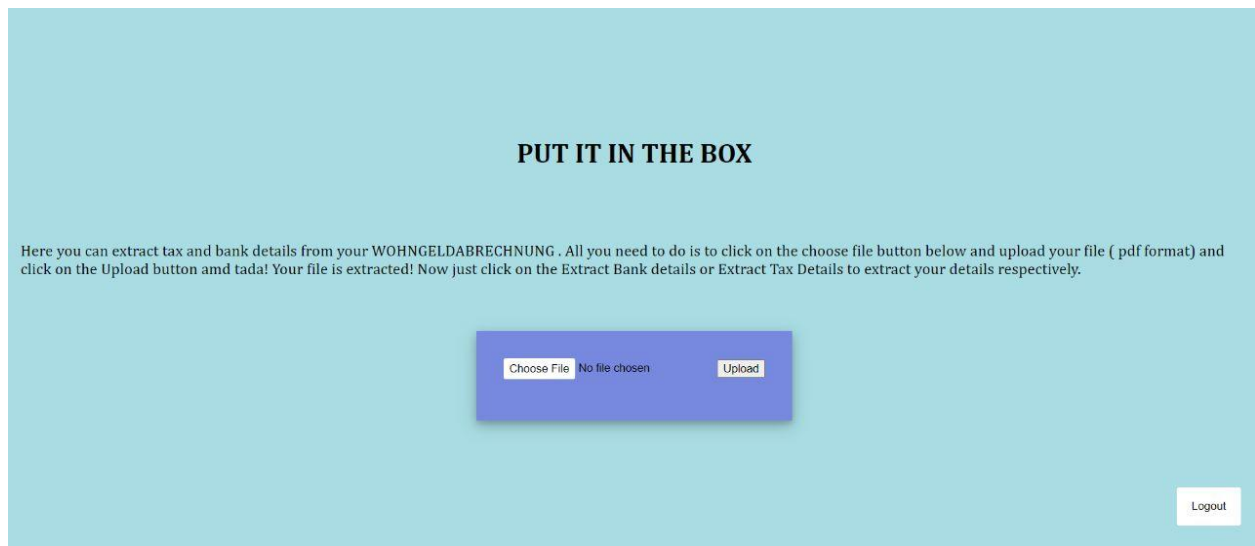
**bank - name , iban , bic , total amount , purpose. Paid to , total expenses**

**expenditure , revenue , development of the maintenance reserve ,result**

### 7.2 Uploading

This is the first step where the user can upload the Wohngeldabrechnung file. Here the user is supposed to upload only .pdf files and other extensions are not supported. The pdf file must be in the structure mentioned above.

The user clicks on the choose file button where they can choose the file from the device and once the upload button is clicked, the file gets uploaded and an instant message is displayed along with the options for extracting the tax details and bank details.



**Figure 6: Uploading a .pdf file**

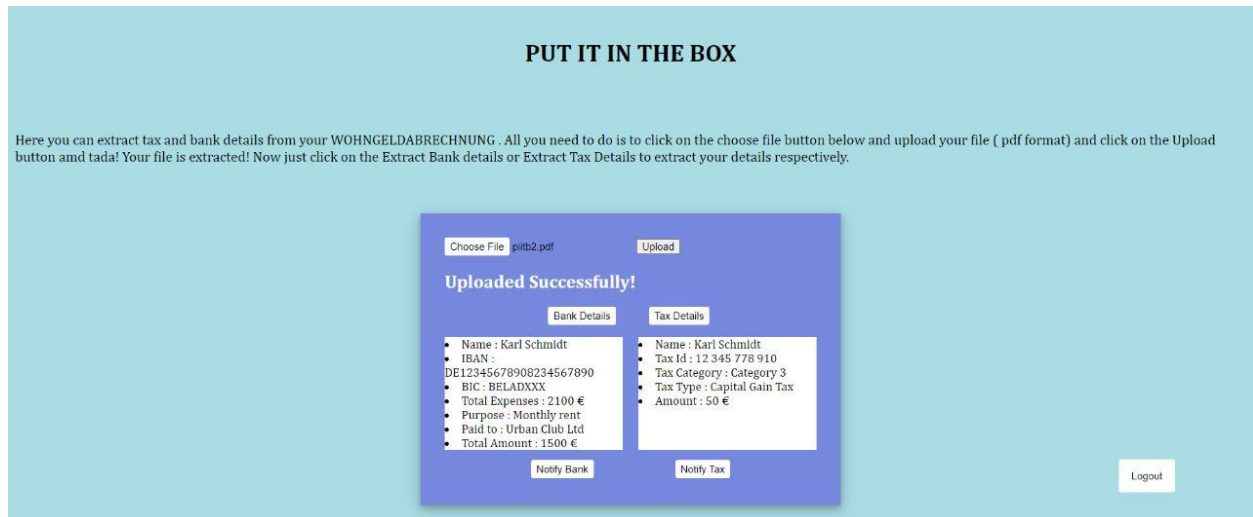
## 7.2 Extraction

After uploading the file , the next step is extraction of bank and tax details. Here We are extracting specific information from a given text using regular expressions. The fields include:

- Name: Name of the customer
- IBAN: IBAN of the customer
- BIC: BIC of the customer
- TotalExpenses: Total expenses for a month
- Purpose: Reason of expenditure
- PaidTo: Name of organization
- TotalAmount: Total amount to be paid
- TaxId: Tax ID of customer

- TaxCategory: Tax Category of the customer
- TaxType: Type of Tax
- Amount: Total Amount to be paid.

After extracting the user can either notify the officers or log out. Log out will redirect to the initial page.



**Figure 7: Extracting the details**

### 7.3 Notifying

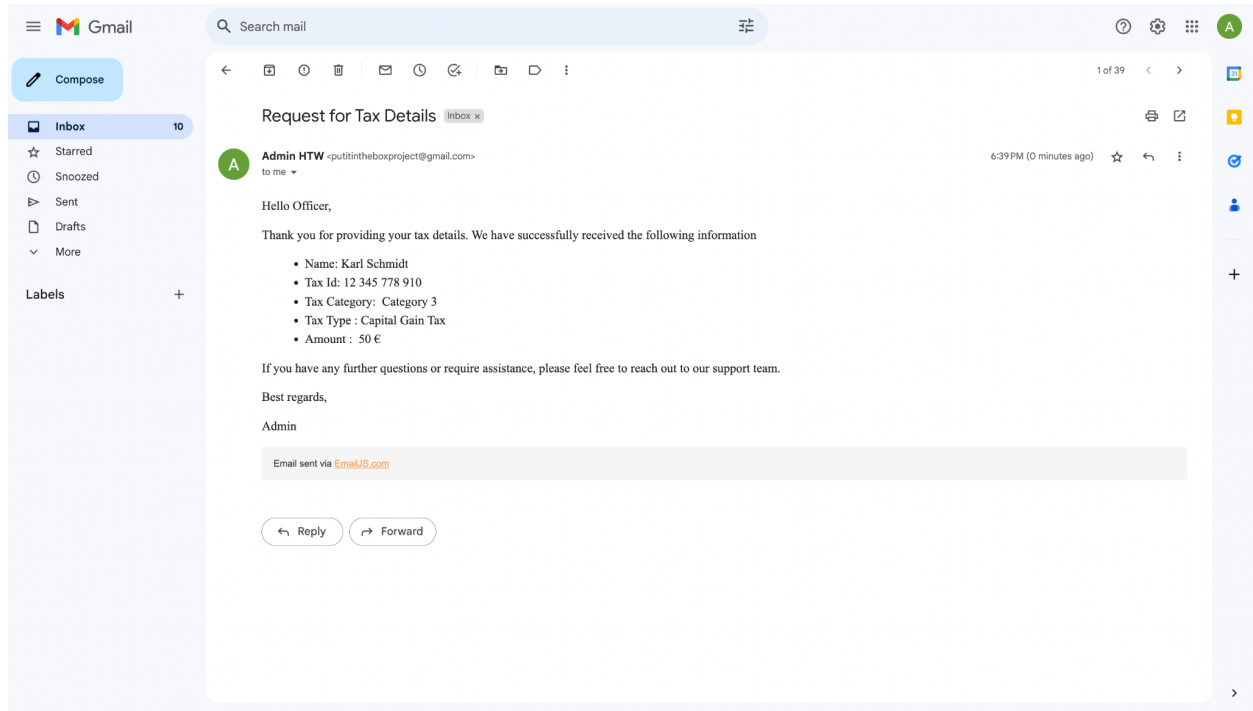
After Extracting the user is allowed to notify the extracted details to the respective officers via mail.

Below details are sent to bank officers:

- Name: Name of the customer
- IBAN: IBAN of the customer
- BIC: BIC of the customer
- TotalExpenses: Total expenses for a month
- Purpose: Reason of expenditure
- PaidTo: Name of organization
- TotalAmount: Total amount to be paid

Similarly below details are sent to Tax officers:

- TaxId: TaxID of customer
- TaxCategory: Tax Category of the customer
- TaxType: Type of Tax
- Amount: Total Amount to be paid.



**Figure 8: Notifying**

## 7.4 Encryption

Since our website stores sensitive user data, it is important to encrypt it even while storing Locally.

### Inner Structure/algorithms:

#### SQLCipher

We are using a third party library called SQLCipher. SQLCipher [1] adds an encryption layer to SQLite [2].

The encryption is pretty strong, using AES-256, a standard in the industry.

The app code contains an encryption key that will be used to encrypt the data. SQLCipher encrypts/decrypts page by page, which improves the performances, also each page is encrypted Differently.

Because SQLCipher is an extension of SQLite, it is compatible with all SQLite functions, it doesn't disturb Room functionings. We can continue using the DAO as usual, nothing special is required.

1. <https://www.zetetic.net/sqlcipher/>
2. <https://www.sqlite.org/index.html>

## APIs

SQLCipher provides a SupportFactory API which can be injected into the Room database to enable encryption. The support factory takes in the encryption key/password as the only parameter.

```
final SupportFactory factory = new
SupportFactory(SQLiteDatabase.getBytes("<Password>".toCharArray()));

INSTANCE = Room.databaseBuilder(
context, PTDatabase.class, "USER_DATABASE")
    .fallbackToDestructiveMigration()
    .openHelperFactory(factory) // enable encryption helper
    .build()
```

## 8. Testing

### 8.1 End to End Testing

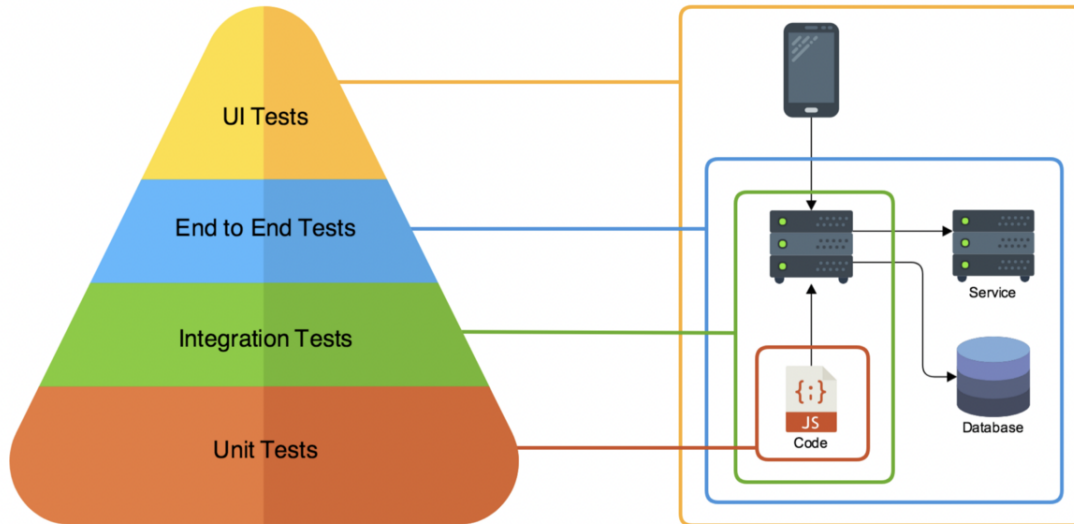


Figure 9: E2E Testing [1]

#### Testing Application:

We have used Puppeteer [2] to test Put it in the box. Puppeteer is a Node.js library that provides a high-level API for controlling Chrome or Chromium browsers. It allows you to automate browser tasks, such as clicking buttons, filling forms, and navigating between pages. Puppeteer is commonly used for end-to-end testing, web scraping, and generating screenshots or PDFs of web pages.

Here are the main components of Puppeteer:

- **Browser:** The `'Browser'` object represents an instance of a headless Chrome or Chromium browser. It can be launched with the `'puppeteer.launch()'` method, which returns a promise that resolves to the `'Browser'` instance. The `'Browser'` object provides methods for creating new pages and managing browser-level operations.
- **Page:** The `'Page'` object represents a single web page within a browser. It provides methods to interact with the web page's content, handle events, execute JavaScript, and perform various actions. You can create a new page using the `'browser.newPage()'` method. The `'Page'` object is where most of the interactions and operations take place.
- **Viewport:** Puppeteer allows you to control the viewport size of a web page. You can set the width, height, and device scale factor to emulate different screen sizes and resolutions. The viewport

1. <https://semaphoreci.com/blog/e2e-testing>
2. <https://pptr.dev/>

- configuration can be specified when creating a new page using the `browser.newPage()` method or by using the `page.setViewport()` method on an existing page.
- Navigation: Puppeteer provides methods to navigate to URLs, reload the page, go back or forward in history, and wait for navigation events to complete. You can use the `page.goto()` method to navigate to a specific URL and the `page.waitForNavigation()` method to wait for the page to finish loading or for a specific navigation event to occur.
- Interactions: Puppeteer allows you to simulate user interactions with web pages, such as clicking elements, typing text, submitting forms, scrolling, and hovering over elements. You can use methods like `page.click()`, `page.type()`, `page.focus()`, and `page.hover()` to perform these interactions.
- Evaluation: Puppeteer enables you to execute JavaScript code within the context of a web page. You can use the `page.evaluate()` or `page.evaluateHandle()` methods to run scripts and retrieve values from the page's DOM or interact with JavaScript functions defined within the page.
- Screenshot and PDF Generation: Puppeteer provides methods to capture screenshots and generate PDF files of web pages. You can use the `page.screenshot()` method to capture a screenshot and the `page.pdf()` method to generate a PDF. These methods offer options to control the output format, quality, page size, and more.

Puppeteer code to test the UI of a web page:

```
const puppeteer = require('puppeteer');

(async () => {
  // Launch a new browser instance
  const browser = await puppeteer.launch();

  // Create a new page
  const page = await browser.newPage();

  // Navigate to the web page you want to test
  await page.goto('https://example.com');

  // Perform UI interactions and assertions
  await page.waitForSelector('h1'); // Wait for the h1 element to appear
  const title = await page.$eval('h1', element => element.textContent); // Get the text
    content of the h1 element
  console.log('Page title:', title);

  const linkCount = await page.$$eval('a', links => links.length); // Count the number of
    <a> elements on the page
  console.log('Number of links:', linkCount);
})
```



```
// Capture a screenshot
await page.screenshot({ path: 'screenshot.png' });

// Generate a PDF
await page.pdf({ path: 'page.pdf' });

// Close the browser
await browser.close();
}) ();
```

### Application's Manual Testing Results

Test case	Test description	Status
Login	Users should be able to successfully login to the website.	Passed
Upload PDF file	Users should be able to upload the pdf file on the website without any error.	Passed
Extract Bank details	Users should be able to extract the bank details after uploading the file.	Passed
Extract Tax details	Users should be able to extract the Tax details after uploading the file.	Passed
Notify via mail	The website should be able to notify the relevant information to the officers.	Passed

**Table 2: Manual testing result.**

## 9. Summary

The project aims to address the challenges associated with Wohngeldabrechnung (housing allowance calculation) by developing a user-friendly website called "PUT IT IN THE BOX." The key features of the website include an advanced technology-based system that extracts relevant information [1] , such as bank and tax details, from Wohngeldabrechnung documents. Users can easily upload their documents, and the intelligent system swiftly analyzes and extracts the necessary information. The website is built using modern web technologies, particularly leveraging React and JavaScript, to create an interactive and dynamic user interface. This enables seamless navigation and efficient data display, ensuring a smooth user experience.

## 10. Conclusion and Future work

In conclusion, the "PUT IT IN THE BOX" project addresses the challenges associated with Wohngeldabrechnung processing by providing a user-friendly website that streamlines the application process for individuals seeking housing benefits and enhances the efficiency for housing authority professionals. By leveraging advanced technology and modern web development practices, the project offers a solution that simplifies data extraction, improves transparency, and facilitates communication between users and housing authorities. The project has the potential to significantly improve the accuracy, speed, and accessibility of the Wohngeldabrechnung process, ultimately benefiting low-income households and ensuring access to adequate housing.

There are several areas for future work and enhancement: further development could focus on integrating the website with existing housing benefit systems used by housing authorities. This would allow for seamless data exchange, reducing manual effort and improving data accuracy. Also Strengthening security measures to protect sensitive user data during the file upload and information extraction processes would be essential. Implementing encryption protocols [1] and following best practices for data privacy would further enhance the platform's reliability. Adding to this collecting feedback from users, including both individuals and housing authority professionals, would provide valuable insights for iterative improvements. Regular updates and enhancements based on user feedback would ensure that the platform continues to meet their evolving needs.

## References

- [1] <https://gitlab.rz.htw-berlin.de/s0585834/pdfproject>