|  |
| --- |
|  |
| Home Link |
| Methods in Software Engineering |
| Project Book |
| **Authors:**  Tslil Aharon - 318320694  Linoy Noe - 316187863  Noam Bassous – 206843237  Date: |
| **‏‏12/06/2024** |

*Abstract*

Following the war, a significant number of people found themselves without shelter, forced to look for temporary accommodation in hotels or various guesthouses. Entire families with children were crammed into small rooms, which created difficulties in managing the daily routine. Finding temporary housing can be a challenging task, and our idea is to successfully deal with this dilemma.

Our platform will connect apartment owners who are willing to donate their properties to evacuees and people who remain homeless, looking for short-term rentals. The system will include a database of available properties, allowing filtering and sorting based on specific parameters.

Each property listing will provide detailed information about nearby amenities such as grocery stores, supermarkets, playgrounds and medical facilities. The system aims to encourage property owners to donate temporary housing through the creation of a competitive rating mechanism that strengthens and promotes social ties and contribution to the community. Feedback and ratings can be provided at the end of each hospitality experience, fostering a sense of community and gratitude. Through this innovative approach, our platform seeks to alleviate the housing crisis and enable a supportive environment for those in need. In the system it will be possible to receive offers according to the personal preferences of the evacuee and it will be possible to rate the owner of the property after the stay according to the hospitality experience.

Table of Contents

[1 INTRODUCTION 6](#_Toc159048112)

[1.1 Problem Description 6](#_Toc159048113)

[1.2 HomeLink Motivation 7](#_Toc159048114)

[1.3 HomeLink Goals 8](#_Toc159048115)

[1.4 Overview of the HomeLink Approach 8](#_Toc159048116)

[1.5 Usage scenarios 8](#_Toc159048117)

[1.5.1 Scenario 1 - Subsets of existing files/directories 8](#_Toc159048118)

[1.5.2 Scenario 2 - Using the same aggregation into a different order of folders 8](#_Toc159048119)

[1.5.3 Scenario 3 - Using different aggregation into new folders 9](#_Toc159048120)

[1.6 HomeLink audience 9](#_Toc159048121)

[1.7 Glossary 9](#_Toc159048122)

[1.7.1 Generic terminology 9](#_Toc159048123)

[1.7.2 HomeLink specific terminology 10](#_Toc159048124)

[2 TECHNOLOGICAL SURVEY 12](#_Toc159048125)

[3 REQUIREMENTS AND SPECIFICATION 13](#_Toc159048126)

[3.1 HomeLink functional requirements 13](#_Toc159048127)

[3.2 HomeLink non-Functional requirements 13](#_Toc159048128)

[3.3 Specification - the scope of the HomeLink project 14](#_Toc159048129)

[3.4 The HomeLink approach 14](#_Toc159048130)

[4 THE ARCHITECTURE 16](#_Toc159048131)

[4.1 HomeLink Data 16](#_Toc159048132)

[4.1.1 The internal and external representations (of files and folders) 16](#_Toc159048133)

[4.1.2 Initial start-up data 16](#_Toc159048134)

[4.2 HomeLink Processes 17](#_Toc159048135)

[4.3 Roles 18](#_Toc159048136)

[4.4 The life-cycle view of a system 18](#_Toc159048137)

[5 SYSTEM DESIGN 21](#_Toc159048138)

[5.1 Data components 21](#_Toc159048139)

[5.1.1 The Internal Representation (InRep) data structures 21](#_Toc159048140)

[21](#_Toc159048141)

[21](#_Toc159048142)

[5.1.2 The External Representation (ExRep) display 22](#_Toc159048143)

[5.2 Process components 22](#_Toc159048144)

[5.3 Communication components (user-computer interaction) 23](#_Toc159048145)

[5.4 Interactions 23](#_Toc159048146)

[5.4.1 State diagram 23](#_Toc159048147)

[5.4.2 Design sequence diagram 25](#_Toc159048148)

[5.5 System Architecture 25](#_Toc159048149)

[6 IMPLEMENTATION 27](#_Toc159048150)

[6.1 Algorithm Design 27](#_Toc159048151)

[6.2 Interfaces 27](#_Toc159048152)

[6.3 Development environment 31](#_Toc159048153)

[6.4 Programming languages 32](#_Toc159048154)

[6.5 Risk Management 32](#_Toc159048155)

[6.6 Exceptions Management 32](#_Toc159048156)

[6.7 Versions Control 32](#_Toc159048157)

[6.8 Project Management 32](#_Toc159048158)

[6.9 Code 33](#_Toc159048159)

[7 SYSTEM VALIDATION 34](#_Toc159048160)

[8 SUMMARY, EVALUATION, CONCLUSIONS AND FUTURE WORK 35](#_Toc159048161)

[8.1 Summary 35](#_Toc159048162)

[8.2 Evaluation and conclusions 35](#_Toc159048163)

[8.3 Future work 35](#_Toc159048164)

[9 REFERENCES 37](#_Toc159048165)

[10 APPENDIX A: Background material 38](#_Toc159048166)

# INTRODUCTION

Our system serves as a platform for finding housing solutions for evacuees and individuals left without shelter for short periods. The platform enables property owners to efficiently and easily publish their listings, providing users with an advanced search and filtering experience to quickly find the most suitable accommodation for them.

## Problem Description

Due to the war, many people have found themselves without shelter or displaced from their homes. Numerous property owners are eager to contribute to their community by renting out their properties as donations for those who have been displaced. This initiative allows property owners to extend a helping hand to those in need, providing temporary housing for those affected by the conflict.

## Home link Motivation

## By offering their properties for rent as a contribution, these property owners aim to support the community and provide assistance in any way they can. This initiative not only provides shelter for those affected by the conflict but also strengthens the sense of community.

## Home Link Goals

Facilitating a streamlined process for evacuees to secure temporary housing during times of crisis is essential. Creating a user-friendly platform that connects evacuees with available rental properties emphasizes ease of use for both property owners and those seeking shelter. Enhancing the search experience for evacuees by providing a fast and efficient platform that precisely matches their specific housing requirements is also crucial. Implementing advanced search and filtering mechanisms allows evacuees to quickly find accommodations tailored to their preferences, such as location, size, and amenities. Furthermore, fostering a sense of community and encouraging property owners to contribute to the well-being of evacuees through their rental properties is vital. Developing a competitive rating system incentivizes property owners to offer their spaces for short-term use, emphasizing community engagement and contribution.

## 1.4 Overview of the Home Link Approach

The project provides free rental apartments for evacuees displaced by the war through a user-friendly website that connects them with property owners willing to offer their properties at no cost. The platform features advanced search and filtering mechanisms, allowing evacuees to find accommodations that meet their specific needs, while property owners can easily list and manage their properties. A rating system recognizes and incentivizes property owners' contributions, fostering a sense of community and support during difficult times.

## Usage scenarios

## 

## Home Link audience

The primary audience for the project includes evacuees displaced by the war who are in urgent need of temporary housing and property owners who are willing to offer their properties for free to support those in crisis.

### 

# 2 TECHNOLOGICAL SURVEY

2.1 Programming Languages and Frameworks

For the development of the project, React is chosen as the primary framework. React is known for its efficiency in building interactive user interfaces and single-page applications. It allows for the creation of reusable UI components, which enhances development speed and maintainability. The declarative nature of React ensures that the UI remains consistent and predictable, which is crucial for a user-friendly platform that connects evacuees with property owners.

2.2 Database Management

The project utilizes MongoDB as the database solution. MongoDB, a NoSQL database, is selected for its flexibility in handling unstructured data and its ability to scale horizontally. This is particularly beneficial for a platform that needs to manage a diverse set of data, including property listings, user profiles, and search queries. MongoDB's schema-less nature allows for rapid iteration and evolution of the data model as the project grows and its requirements change.

2.3 Additional Technologies

To further improve the platform's functionality and user engagement, technologies such as RESTful APIs for seamless data communication and \*\*authentication mechanisms\*\* for secure user access are employed. These technologies contribute to a robust and scalable system capable of handling a large number of users and interactions effectively.

# 3 REQUIREMENTS AND SPECIFICATION

**Functional** requirements of the project are the necessary conditions that relate to the functionality of the system. **Non-functional requirements** of the project are requirements which do not relate to functionality but to aspects that influence the manner in which the functionality is provided.

## Home Link functional requirements

* + 1. User Registration and Authentication:
* The system should allow users to create accounts, providing necessary information for authentication.
* Users should be able to log in securely using credentials.  
  + 1. User Profiles:
* Each user, whether an evacuee or a property owner, should have a profile with editable personal information.
* Profiles should include relevant details such as contact information, preferences, and, for property owners, property details.  
  + 1. Property Listing:
* Property owners should be able to upload detailed information about their available accommodations, including capacity, amenities, and nearby facilities.
* The system should support image uploads to enhance property presentation.  
  + 1. Property Search and Reservation:
* Evacuees should be able to search for available properties based on criteria like location, capacity, and amenities.
* The system should provide a reservation feature allowing evacuees to book properties for specified durations.  
  + 1. Feedback and Rating System:
* Users, especially evacuees, should have the ability to provide feedback on their stay, including cleanliness, amenities, and overall satisfaction.
* Property owners should receive ratings based on tenant feedback, contributing to a community-driven reputation system.  
  + 1. Notification and Communication:
* The system should send automated notifications to users, informing them of reservation confirmations, feedback requests, and other relevant updates.
* An internal messaging system should facilitate communication between evacuees and property owners.
  + 1. Reservation Management:
* Property owners should have a dashboard to manage reservations, including viewing upcoming bookings, confirming reservations, and marking property availability.

## Home Link non-Functional requirements

The following is a list of non-functional requirements:

* + 1. Performance:
* The system should be able to handle a scalable number of users and properties, ensuring optimal performance during peak usage times.
* Property search and reservation processes should be responsive, with low latency.  
  + 1. Scalability:
* The system architecture should support scalability to accommodate a growing number of users, properties, and concurrent transactions.
* Scaling mechanisms should be implemented to handle increased load gracefully.  
  + 1. Reliability:
* The system should have high availability, minimizing downtime for maintenance or unexpected issues.
* Data integrity should be maintained, and the system should be resilient to server failures.  
  + 1. Security:
* Secure authentication mechanisms should be implemented to protect user accounts and sensitive information.
* The system should adhere to industry-standard security practices to prevent unauthorized access and data breaches.  
  + 1. Usability:
* The user interface should be intuitive and user-friendly, catering to users with varying levels of technical expertise.
* Navigational elements should be clear, and the system should provide contextual guidance.  
  + 1. Compatibility:
* The system should be compatible with a variety of devices and browsers to ensure a seamless user experience.   
  + 1. Data Backup and Recovery:
* Regular backups of user data and system configurations should be performed to prevent data loss.
* The system should have mechanisms in place for quick data recovery in case of unforeseen events.  
  + 1. Regulatory Compliance:
* The system should comply with relevant data protection regulations and privacy laws.
* Adherence to legal standards and guidelines should be a priority in the system design and operation.

* + 1. Response Time:
* The system should maintain acceptable response times for critical functions such as property search, reservation, and feedback submission.
* Users should experience minimal delays in accessing and interacting with the system.

## Specification - the scope of the Home Link project

## Our project connects evacuees with property owners offering free rental accommodations. The platform allows secure user registration and login, with profiles for both evacuees and property owners. Property owners can list detailed property information and images. Evacuees can search for properties based on criteria like location and amenities, and book them for specific durations. A feedback and rating system will enhance trust and community. Notifications and messaging will keep users informed and facilitate communication. Property owners can manage reservations through a dedicated dashboard.

## Non-functional requirements ensure the system is scalable, reliable, secure, and user-friendly, with regular data backups and compliance with data protection regulations.

## The Home Link approach

The proposed solution approach for the temporary shelter system encompasses a secure user authentication and registration system, allowing individuals like Shani and Eli to access the platform. The system facilitates property listing and search functionalities, ensuring evacuees can easily find suitable accommodations, while property owners, like Eli, can seamlessly upload and manage their listings. A robust reservation system, coupled with a comprehensive feedback and rating mechanism, enhances user experience. Personalized user profiles and dashboards, along with effective notification and communication features, contribute to a user-friendly interface.

|  |
| --- |
|  |

# THE ARCHITECTURE

To explain the architectures, we will introduce several models later on. These models will help illustrate data transfer methods, highlight the system's key features, and clarify the interactions between the system, users, and administrators.

## Home Link Data

The system's database is managed using MongoDB. This database stores all relevant information, including user profiles, property details, and reservation data. MongoDB's flexibility and scalability ensure that the platform can efficiently handle the growing amount of data and user interactions, providing a reliable foundation for connecting evacuees with available accommodations.

### The internal and external representations

* **Internal Representation**

The internal representation involves the backend infrastructure, including the MongoDB database for storing user profiles, property details, reservations, and feedback. It also includes server-side logic for authentication, data processing, and search algorithms.

* **External Representation**

The external representation consists of the user interface built with React, allowing users to create accounts, manage profiles, search for properties, and make reservations. It focuses on visual design, usability, and features like notifications and messaging to facilitate user interaction.

### Initial start-up data

The initial start-up data refers to the foundational information and configurations required to launch and operate the Home Link platform effectively. This data includes:

1. User Accounts: Pre-created user accounts for testing purposes, including profiles for both evacuees and property owners.

2. Property Listings: Sample property listings to demonstrate the system's functionality, complete with details such as location, capacity, amenities, and images.

3. System Configuration: Settings and preferences that define how the system operates, including user roles, permissions, and security protocols.

4. Sample Reservations and Feedback: Example reservations and feedback entries to test and showcase the booking and rating systems.

5. Notification Templates: Predefined templates for automated notifications, such as reservation confirmations and feedback requests.

|  |
| --- |
|  |

## Home Link Processes

The Home Link platform integrates several key processes to provide a comprehensive and user-friendly experience for both evacuees and property owners.

* **User Registration and Authentication**: Users create accounts and log in securely.
* **Profile Management**: Users (evacuees and property owners) manage personal and property information.
* **Property Listing**: Owners upload and manage details of available accommodations.
* **Property Search and Reservation**: Evacuees search for and book properties based on specific criteria.
* **Feedback and Rating**: Users provide and view feedback on properties and stays.
* **Notification and Communication**: Automated notifications and messaging facilitate updates and communication between users.
* **Reservation Management**: Owners manage bookings and property availability through a dedicated dashboard.

|  |
| --- |
|  |

## Roles

**Evacuees**: Individuals seeking temporary housing due to displacement. They use the platform to find, book, and provide feedback on properties.

**Property Owners**: Individuals offering free accommodations to evacuees. They list and manage their properties, respond to bookings, and receive feedback.

These roles ensure the platform effectively connects those in need of housing with those offering support.

## The life-cycle view of a system

The life-cycle view of the Home Link system includes planning and requirements gathering, where objectives and user needs are defined. This is followed by system design, developing the architecture and interface. Implementation involves coding and integrating components. Testing ensures the system meets requirements and is bug-free. Deployment makes the system accessible to users. During operation and maintenance, performance is monitored, and updates are made. Evaluation involves collecting user feedback for improvements. Eventually, the system is retired when it is no longer needed or replaced, ensuring proper data migration and decommissioning.

# 

# SYSTEM DESIGN

This section discusses the design of Home Link; it will explain the mapping of the architecture and its elements to the components that will implement and animate them.

## 

## Data components

### The Home Link project has two major data entities: Users and Properties. User information is used for registration, authentication, and profile management. Property data, including details like location, capacity, amenities, and images, is retrieved from the database and displayed on the user interface, both in a list format and on a map, to facilitate easy search and reservation by evacuees.

## Data Diagram:

## Process components

The process components of the Home Link project include several key operations that ensure the system functions smoothly and efficiently:

1. User Registration and Authentication: This process involves creating new user accounts, verifying identities, and enabling secure login for both evacuees and property owners.

2. Profile Management: Users can edit and update their profiles, including personal information and property details for owners.

3. Property Listing: Property owners upload and manage information about their available accommodations, including adding images and specifying amenities.

4. Property Search and Reservation: Evacuees search for properties based on criteria such as location and capacity, and make reservations for their preferred accommodations.

5. Feedback and Rating: Users provide feedback and ratings for properties, which helps maintain a reputation system for property owners.

## Communication components (user-computer interaction)

## Interactions

### The Home Link project employs a React-based web application to ensure smooth user-computer interaction. The user interface is clean and responsive, guiding users through registration, property search, and reservation processes with ease. Users can manage their profiles and update personal information via a user-friendly dashboard. Interactive search and filter tools allow evacuees to find properties that meet their specific needs, with results displayed in both list and map views. Detailed property pages provide comprehensive information, including descriptions and images. The reservation system enables easy booking with clear confirmation messages. Additionally, users can submit feedback and ratings, contributing to the platform's reputation system.

### State diagram

|  |
| --- |
|  |

### Design sequence diagram

A diagram of a project

Description automatically generated

## 

## A diagram of a software project Description automatically generated with medium confidenceSystem Architecture

# 

# 6 IMPLEMENTATION

## The implementation of the Home Link project involves several key steps to build and deploy the platform effectively:

## Frontend Development: Using React, the user interface is developed to be responsive and user-friendly. This includes creating components for user registration, profile management, property listing, search and filtering, and reservation processes.

## Backend Development: A robust backend is built using server-side technologies that manage user authentication, data processing, and business logic. This includes integrating with MongoDB to store user profiles, property details, reservations, and feedback.

## Database Management: MongoDB is configured to handle data storage and retrieval efficiently. The database schema is designed to accommodate user and property data, ensuring scalability and flexibility.

## API Integration: RESTful APIs are developed to facilitate communication between the frontend and backend. These APIs handle requests for user actions such as logging in, searching for properties, booking accommodations, and submitting feedback.

## Interfaces

## 

This section will present the user interface, which is the web-based application ProFiler. The section will include screenshots of the different screens of the GUI.

**First screen – default view**: the first screen presented is the default view (left side of the screen) of the files and folders as they are stored in the File System. The user can open or close levels of folders. This view will be present at all time for the user to use (Figure 6-3).

|  |
| --- |
|  |

Figure ‎6‑4: Default view (left side)

**Guide**: according to the user’s cookies, if this is the first time the user uses ProFiler a guide will be presented to him, explaining the main features and sections of the system. The guide will be always available using the ‘?’ button (Figure 6-4).

|  |
| --- |
| C:\Users\tcohen\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.Word\Screen Shot 2015-08-04 at 10.45.17 PM.PNG |

Figure ‎6‑5: Guide button

**ProFiler sections:** profiler is divided to 3 sections: intermediate, working stations which the user can create, re-structure and search and the default view (left side) of the files and folders as they are stored in the File System (Figure 6-5).

|  |
| --- |
| Screen Shot 2015-08-04 at 10 |

Figure ‎6‑6: ProFiler sections

**ProFiler features:** profiler provides the user to drag & drop files, folders and trees, save and restore trees which created by the user (Figures 6-7 and Figure 6-8).

|  |
| --- |
| Screen Shot 2015-08-04 at 10 |

Figure ‎6‑7: Drag & Drop feature

|  |  |
| --- | --- |
| Screen Shot 2015-08-04 at 10 | Screen Shot 2015-08-04 at 10 |

Figure ‎6‑8: ‘Save tree’ button and ‘open and choose tree’ button

**Intermediate station**: in this section of the screen, the user can search\filter files and folders by typing the relevant keywords (Figure 6-9).

|  |
| --- |
| Screen Shot 2015-08-04 at 10 |

Figure ‎6‑9: Search/filter files and folders

**Working station**: the user can drag and drop files and folders (along with their content) to this section of the screen to build trees (Figure 6-10).

|  |
| --- |
| Screen Shot 2015-08-04 at 10 |

Figure ‎6‑10: Working station

**Saving a tree**: the user can save a tree for future work by using the save button. While saving the tree the user can name the tree and add comments (Figure 6-11).

|  |
| --- |
| Screen Shot 2015-08-04 at 10 |

Figure ‎6‑11: Saving a tree

**Restore saved tree**: the user can restore a tree he previously saved and continue modifying it. When the user opens a saved tree additional information is displayed on the right side of the screen (Figure 6-12 and Figure 6-13).

|  |
| --- |
| Screen Shot 2015-08-04 at 10 |

Figure ‎6‑12: Restore saved tree

|  |
| --- |
| C:\Users\tcohen\Downloads\Screen Shot 2015-08-04 at 10.53.45 PM.png |

Figure ‎6‑13: Restored tree

## Development environment

## Our development environment for the Home Link project is set up to make it easy for our team to work together. We use Visual Studio Code for coding, which works well with JavaScript and backend technologies. Git and GitHub help us manage version control and host our code, allowing the team to collaborate and keep track of changes. For the frontend, we build with React, using Create React App to set up our project. On the backend, we use Node.js and Express.js to build our server-side application. We store our data in MongoDB and manage it with tools like MongoDB Compass

## Programming languages

For the Home Link project, we use a combination of programming languages to build a robust and efficient platform.

* **JavaScript**
* **HTML/CSS**.
* **Mongo DB**
* **React**

This combination of languages enables us to create a seamless, full-stack development environment that supports our project's goals and requirements.

## Risk Management

* + Inadequate Community Contribution:  
    Property owners may not actively contribute or may withdraw from the program.  
    Solution: Implement a competitive rating system to encourage positive contributions. Recognize and reward property owners who consistently contribute.
  + User Disputes:  
    Disputes between evacuees and property owners regarding bookings or property conditions.  
    Solution: Establish clear terms of use and guidelines.
  + Verification of user information:  
    Citizens who will pretend to be evacuees in order to get a free "vacation".  
    Solution: Checking the residential address of the evacuees when booking the property in order to verify that they are indeed evacuees.

## Exceptions Management

This section includes erroring and exceptions handling.

## Versions Control

You may include here print screens of your versions control processes, or links to open version controls you used over the year.

## Project Management

This section includes tasks and time management. הגאנט שלכם, חלוקת עבודה ביניכם וכל ניהול הפרויקט כולל חלוקה לאלפא ולסוף פרויקט

## 

## Code

This section includes links to GitHub or any other code repository. If you deliver the code in any other way, this section should include directions regarding where the code is and how to test it.

You may show examples of main code sections

1. Code to GitHub
2. Code to your recording of using the system
3. Code to your working system online
4. Readme if needed

# 

# SYSTEM VALIDATION

פה עליכם להציג את דוח הוולידציה, כפי שהוצגה בפני השופטים, כולל פסקת הסבר, מה שיניתם, מסקנות ותיאור התהליך. במידה והמשכתם עם עוד וולידציה בקיץ, ניתן להוסיפה.

# SUMMARY, EVALUATION, CONCLUSIONS AND

# FUTURE WORK

## Summary

Working with files, folders or any other information source has a real purpose in our world; any information can serve different purposes at different times for different people. ProFiler is a PIM tool that provides the means to create, manage, organize and maintain the points of view the user needs. A user may want to look at files according to one classification and then look at the same set or a subset of those files using another classification.

ProFiler was developed in order to address this issue - to provide the user with the ability to create and modify different points of views of the Files/Directories File System. ProFiler’s architecture is designed to apply unlimited transformation on any set of hierarchical files and folders and to store the results for later use. In short - ProFiler is a first step in developing a personal information management tool that is focused on the representation of the different points of view.

The second generation of ProFiler should include several extensions to further assist people who use this application (see sub-section 7.2 - Future work). The basic issues relate to the fact that ProFiler creates alternative views of the File System. Those views and the original data in the File System can get out of step and become inconsistent.

## Evaluation and conclusions

No extensive field studies were conducted with ProFiler and this hinders a thorough and meaningful comparison of traditional browsers with ProFiler. However, from an individual point of view, ProFiler works well - allowing the user to construct multiple virtual views based on the initial default view of the File System (FS). Virtual views can be constructed by modifying the structure of the folders, by adding new folders with links among them and by populating them with some or all of the existing files from the FS. Using the Browser based GUI allows the user to exploit the 'drag&drop' mode of interaction in an intuitive manner.

ProFiler also allows saving a virtual view and retrieving it at a later point in time making the desired views persistent. In that respect ProFiler answers all the requirements agreed on in the specification phase of the project (Section 4.1).

When using the system, however, it becomes patently clear that an aspect that was left out of the project due to time and man-power constraints makes the system's usefulness limited and perhaps even hazardous. We refer to the following point from Section 4.2:

* **FS changes & consistency checks**: any changes in the FS caused by the user (through the use of the default FS browsers) which affect the views derived by ProFiler should be conveyed to the user. The ProFiler **will not be notified** of changes in the FS and the changes made in ProFiler will have no effect on the FS.

In retrospect, it appears to be an essential property of such a system since the changes to the FS may make the virtual views inconsistent and thereby misleading.

The decision to design and implement the system as a web application had some advantages and disadvantages. The advantage was the ease of GUI development providing a flexible system with an impressive interface. The main disadvantages are that there is a need for a local server and it requires an installation process. If such a system was to be built again in the future – we suggest looking carefully at the available technologies and considering building it as a stand-alone application based on a language such as Java, C#, Python, PHP, etc. without the use of a browser and a server.

## Future work

There are several issues which the current version of ProFiler does not deal with and that should be addressed if ProFiler is further developed:

* **Adding attributes**: Adding attributes to a file and adding the ability of sorting/structuring the alternative views according to those attributes. These attributes include: name, created date, comments, link to other files and classification.
* **Saving transformations**: saving of Transformation to be able to apply them (semi) automatically at a later stage. When there is a change in the File System the user can still apply the saved transformation on the update structure (Figure 7-1).
* **Open more types of files**: currently, the application can open WORD documents when they are double-clicked. In order to use ProFiler as a useful file browser that can be used more efficiently, the user should have the ability to open more types of files from the application (such as Doc, PPT, PNG etc.).
* **Modify the GUI**: in order to make ProFiler more popular, the GUI should be modified to have a Windows Explorer look and feel.
* **Consistency checks**: check for inconsistencies that arise from changes made to the File System by the default browser. Should inconsistencies be reflected in the ProFiler data/view and if so – how? Should the changes be made automatically or should the user be notified of any change – for example, when the user is restoring saved views (Figure 7-2.
* **ProFiler’s modifications:** allow updates in ProFiler data to be reflected in the File System (Figure 7-3). The consequences of such ability are not clear and therefore serious consideration is required before attempting to add it.

|  |
| --- |
| File System  ProFiler View  Transformation  Save  Apply |

Figure ‎7‑1: Saving transformation and being able to apply them later semi-automatically

|  |
| --- |
| Updates  File system  ProFiler View  Modifications |

Figure ‎7‑2: ProFiler views should be made aware of changes made to the File System - keeping things consistent may prove tricky

|  |
| --- |
| Updates  File system  ProFiler View  Modifications |

Figure ‎7‑3: Allowing specified modifications performed in ProFiler to affect the File System – an open question

# REFERENCES

**[Barreau 95]** Barreau, D. and Nardi, B. “*Finding and Reminding: File Organization from the Desktop*”, SIGCHI Bulletin, 27(3), July 1995.

[**Bower 15**] "*Install Bower packages*", 2015, <https://www.npmjs.com/package/gulp-bower>

**[Dourish 99]** Dourish, P. W., Edwards, K., LaMarca K. and Salisbury, M. "*Presto - An Experimental Architecture for Fluid Interactive Document Spaces*", Computer Science Laboratory, 1999.

**[C# 15]** "How to: Get Information About Files, Folders, and Drives (C# Programming Guide)" , 2015, [*https://msdn.microsoft.com/en-GB/library/6yk7a1b0.aspx*](https://msdn.microsoft.com/en-GB/library/6yk7a1b0.aspx)

**[Explorer 15]** *"Windows Explorer or File Explorer*", 2015, [*https://en.wikipedia.org/wiki/File\_Explorer*](https://en.wikipedia.org/wiki/File_Explorer)

[**Express** 15] *“Fast, un opinionated, minimalist web framework for Node.js”*, 2015, <http://expressjs.com/>

[**File Manager 15**] "*File manager*", 2015, <https://en.wikipedia.org/wiki/File_manager>

**[Gulp 15]** *“Automate and enhance your workflow for node.js”,* Wikipedia, 2015, <http://gulpjs.com/> and <https://www.npmjs.com/package/gulp-bower>

**[Node.js 14]** *"Node.js v0.12.7 Manual & Documentation"*,2014, <https://nodejs.org/api/>

**[Multi-tier 15]** *"Multi-Tier Architecture"*, 2015, <https://en.wikipedia.org/wiki/Multitier_architecture>

**[Mac Finder 15]** "*Switch Basics: Moving from Windows Explorer to the Finder*", Mac Finder, 2015, <https://support.apple.com/en-gb/HT201748>

**[Mac Finder 14]** "*Mac Finder*",2014,<https://en.wikipedia.org/wiki/Finder_(software)>

**[PhpStorm 15]**  *"Lightning-smart PHP IDE*", 2015, <https://www.jetbrains.com/phpstorm/>

**[Nautilus 15]**  *"Nautilus - Default File Manager in Ubuntu"*, 2015, <https://help.ubuntu.com/community/DefaultFileManager>

[**PIM 15**] "*Personal Information Management*", Wikipedia, 2015, <https://en.wikipedia.org/wiki/Personal_information_management>

**[Q-Dir 15]**  *Q-Dir website*, 2015, <http://www.softwareok.com/?seite=Freeware/Q-Dir>

**[Voit 09]** Voit, K., Andrews, K. & Slany, W., "*Why Personal Information Management (PIM) Technologies Are Not Widespread and What to do About It*", PIM 2009

# 

# APPENDIX A: Background material

The **File Explorer** [Explorer 15] was introduced with Windows 95 (as Windows Explorer). It is a graphical file management utility for the Windows operating system. The goal of the file management utility is to allow users to conveniently work with the files and directories located on any connected disk. When a file is selected within the Windows/File Explorer interface, users can select from a variety of context-sensitive actions such as renaming, moving, copying, publishing, emailing, printing or deleting the file. Entire folders can also be opened, explored, shared, copied, cut, pasted, moved or deleted. Files can be searched or dragged and dropped between folders, and folders can be moved into other folders and disk drives as desired.

The **Mac Finder** [Mac Finder 14 & Mac Finder 15] is the default file manager and graphical user interface shell used on all Macintosh operating systems. The Finder uses a view of the file system that is rendered using a desktop metaphor; that is, the files and folders are represented as appropriate icons.

**Nautilus** is the default file manager in Ubuntu [Nautilus 15].