Lab 1 – Thought Locker Product Description

Nathan Greaney
Old Dominion University
CS411W
Professor James Brunelle
15 September 2023
Version 2 (Draft)
Table of Contents

1 Introduction 3

2 Product Description 4

   2.1 Key Product Features and Capabilities 4

   2.2 Major Components (Hardware/Software) 5

3 Identification of Case Study 7

4 Product Prototype Description 8

   4.1 Prototype Architecture (Hardware/Software) 8

   4.2 Prototype Features and Capabilities 9

   4.3 Prototype Development Challenges 11

5 Glossary 12

6 References 16

List of Figures

Figure 1: Thought Locker Major Functional Components Diagram 6

Figure 2: Thought Locker Prototype Major Functional Components Diagram 9

List of Tables

Table 1: Thought Locker RWP vs. Prototype Features 10
1 Introduction

Dementia is a progressive neurological disorder that affects cognitive functions such as memory, language, and reasoning. As the global population ages, the incidence of dementia is on the rise, and it is becoming an increasingly pressing public health issue. With one in ten Americans aged 70 and older affected by some form of dementia, and with 50.4% of cases being mild and 30.3% moderate, the need for a solution to help these individuals is more pressing than ever ("Fact sheet," n.d.; "Half," n.d.). This rise in dementia has significant implications for healthcare systems and society, making it essential to understand the factors contributing to the increase and to develop strategies for prevention, early diagnosis, and improved care and support.

Dementia typically progresses through several stages, each with its own set of symptoms and challenges. The stages of dementia include mild, moderate, and severe, with the symptoms becoming progressively more severe at each stage. In the early stages of dementia, individuals may experience mild cognitive impairment that does not significantly impact their daily life. However, as the condition progresses, they may experience more pronounced symptoms such as memory loss, confusion, and difficulty with daily tasks. In the later stages of dementia, individuals may lose the ability to communicate, walk, or perform basic tasks independently, and may require round-the-clock care and support. Understanding the different stages of dementia is essential for healthcare professionals, caregivers, and family members to provide appropriate care and support to people living with the condition. By identifying the stage of dementia, caregivers can develop personalized care plans that focus on managing symptoms, promoting a healthy lifestyle, and maintaining quality of life.
As the number of individuals diagnosed with dementia continues to increase, it is essential to provide a solution that allows individuals to maintain their independence for as long as possible. There is currently no cure for dementia, but various treatment and management options can help alleviate symptoms and improve quality of life for people living with the condition. These solutions include medications, lifestyle changes, and therapies, as well as providing support and care for people with dementia and their families. With Thought Locker, individuals with mild to moderate dementia can receive the support they need, while still maintaining some level of independence. This reduces the stress on caregivers and family members who would otherwise be required to provide constant attention and care.

2 Product Description

Thought Locker is an application that provides various features to assist individuals with mild to moderate dementia in their daily living activities. It offers reminders, item location assistance, and contact assistance, allowing dementia sufferers to manage their daily tasks effectively. Thought Locker provides analytics that can be shared with caregivers, enabling them to monitor the individual's progress and provide appropriate support as needed. This monitoring feature can help detect changes in mood and provide insights into the individual's mental state.

2.1 Key Product Features and Capabilities

Thought Locker is designed to help individuals with mild to moderate dementia with their daily living activities. It is compatible with smartphones, tablets, and other devices on Android and iOS operating systems. Users sign up for an account using their email address. Authentication is verified through username and password or biometric data. Thought Locker includes a calendar feature that allows users to schedule reminders and customize settings for reminders and notifications. Users also have access to item finders, enabling them to locate
commonly misplaced items. Thought Locker utilizes sensors to track item location, and caregivers have access to motion sensors to detect activity in the home.

Thought Locker also provides daily reminders and notifications, ensuring that users do not miss important activities or appointments. Additionally, family members or caregivers can monitor patient activities through the patient monitoring feature, which provides an insight into how the patient is doing as far as successful/unsuccessful task completion rates, item finding, and sensor triggers. Thought Locker is customizable, allowing family members or caregivers to tailor the application for each patient's unique needs. Patients can also control some settings to maintain a sense of independence.

2.2 Major Components (Hardware/Software)

The major components of Thought Locker, as illustrated in Figure 1, include hardware components, consisting of a mobile device with internet access, an application server, and a database server, and software components, consisting of programming language, testing framework, repository, APIs, version control, issue tracking, continuous integration, and containerization.

[ This space left intentionally blank. ]
Thought Locker requires a mobile device (either Android or iOS) with internet access to operate. An application server is needed to host Thought Locker. A database server is needed to store and manage user data. The backend server for Thought Locker is hosted on AWS, providing secure and reliable access to the application server and database server. Thought Locker uses PostgreSQL and MongoDB as its databases, allowing for efficient storage and retrieval of user data. It is developed using the JavaScript language, specifically the React and Node.js frameworks. Thought Locker is tested using the Jest testing framework to ensure a bug-free and reliable application. Code will be stored in a GitHub repository, providing easy access and collaboration for developers.

Thought Locker utilizes React-native APIs for Contacts and BLE manager functionality, enabling integration with other applications and services. It also uses the Google Maps Geolocation API for location-based services. Thought Locker’s code is version-controlled using GitHub, allowing for easy tracking of changes and updates. Issues and bugs are tracked using
GitHub's issue tracking functionality, ensuring quick identification and resolution of any issues. Also, Thought Locker is continuously integrated using GitHub's continuous integration functionality, ensuring that the application is always up-to-date and bug-free. Thought Locker is containerized using Docker, enabling efficient deployment and scalability of the application.

3 Identification of Case Study

Thought Locker is a versatile application designed to help individuals with mild to moderate dementia and their caregivers/family members to manage the challenges of the disease. It offers a range of features that cater to different needs, and its prototype will showcase these features in real-life situations. Thought Locker offers a way to locate misplaced items, remind patients of medication and appointments, provide patients with the ability to contact a caregiver when needed, and offer a way for caregivers to monitor patient habits using analytics. In addition to these primary user groups, other stakeholders that could benefit from Thought Locker include medical facilities, insurance companies, and Alzheimer's research groups based on the anonymized data collected.

The case studies are broken down into two groups: individuals with dementia and caregivers. Individuals with dementia are further broken down into two groups: mild and moderate. One scenario is a person with mild dementia who lives on their own. They have total access to all the features of Thought Locker. They add, delete, and modify items to be found, tasks to be completed, and maintain their contact list. Another scenario is a person with moderate dementia who lives with their spouse. The person with dementia has limited access to the features of Thought Locker. They can only find items, complete tasks, and make calls. Their spouse, acting as a caregiver, has total access to all the features of Thought Locker. They assist the person with dementia to add, delete, and modify items to be found, tasks to be completed,
and maintain the contact list. In addition, the spouse can monitor sensors placed around the house, such as doors, windows, gas appliances, and electric switches/outlets, to ensure the safety of the individual with dementia. The spouse can track the individual’s dementia progression with Thought Locker’s analytic suite. They are also able to save reports and notes about their loved one.

4 Product Prototype Description

Thought Locker’s prototype is designed to support individuals with dementia, their families, friends, and caregivers. It aims to enhance the well-being of dementia patients by promoting their independence, ultimately reducing stress for those involved. While the prototype may not have all the planned real-world features, it effectively demonstrates its potential by showcasing core functions that empower dementia patients and provide peace of mind to caregivers.

4.1 Prototype Architecture (Hardware/Software)

Thought Locker's prototype architecture, as seen in Figure 2, closely resembles the envisioned real-world MFCD but makes strategic simplifications and uses simulated data to expedite development and focus on core functionalities. These decisions are made to efficiently demonstrate the platform's potential and to gather essential feedback from users and stakeholders.

[ This space left intentionally blank. ]
The Thought Locker has specific hardware and software requirements to ensure its optimal functionality. For hardware, it is compatible with mobile phones or tablets operating on the Android platform, as well as desktops or laptops running Linux, MacOS, or Windows. In terms of software, it primarily utilizes languages such as Javascript, React, and Node.js for development. Notably, it employs MySQL as its preferred database system instead of PostgreSQL/MongoDB and runs on Amazon Web Services (AWS) as its back-end server. Testing is performed using Jest, and API integrations include Reactive Native and the Google Maps Geolocation API. Additionally, the platform relies on Git/Github for version control and CI/CD processes. Please note that iOS will not be supported at this time, and Android is the chosen mobile operating system for Thought Locker.

4.2 Prototype Features and Capabilities

In the development of the prototype for Thought Locker, it's important to acknowledge that not all aspects of the envisioned final product will be fully realized in this initial version, as
illustrated in Table 1. The decision to prioritize certain features for full implementation while having partial implementation or even eliminating others entirely is a strategic one driven by several considerations: resource constraints, proof of concept, user-centered approach, interactive development, technical challenges, and risk mitigation.

Table 1

*Thought Locker RWP vs. Prototype Features*

<table>
<thead>
<tr>
<th>ACCOUNT MANAGEMENT</th>
<th>Patient</th>
<th>Caregiver</th>
<th>RWP</th>
<th>Prototype</th>
</tr>
</thead>
<tbody>
<tr>
<td>Account Creation</td>
<td>✓</td>
<td>✓</td>
<td>Full</td>
<td>Full</td>
</tr>
<tr>
<td>Account Deletion Verification</td>
<td>✓</td>
<td>✓</td>
<td>Full</td>
<td>Full</td>
</tr>
<tr>
<td>Login / Authentication</td>
<td>✓</td>
<td>✓</td>
<td>Full</td>
<td>Full</td>
</tr>
<tr>
<td>User Profile Management</td>
<td>✓</td>
<td>✓</td>
<td>Full</td>
<td>Full</td>
</tr>
<tr>
<td>User Location Information</td>
<td>✓</td>
<td>✓</td>
<td>Full</td>
<td>Partial: Location data will be simulated</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>USER INTERFACE</th>
<th>Patient</th>
<th>Caregiver</th>
<th>RWP</th>
<th>Prototype</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task Scheduling</td>
<td>✓</td>
<td>✓</td>
<td>Full</td>
<td>Full</td>
</tr>
<tr>
<td>Task Completion</td>
<td>✓</td>
<td>✓</td>
<td>Full</td>
<td>Full</td>
</tr>
<tr>
<td>Task Deletion Verification</td>
<td>✓</td>
<td>✓</td>
<td>Full</td>
<td>Full</td>
</tr>
<tr>
<td>Item Locator</td>
<td>✓</td>
<td>✓</td>
<td>Full</td>
<td>Partial: Location data will be simulated</td>
</tr>
<tr>
<td>Item Registration</td>
<td>✓</td>
<td>✓</td>
<td>Full</td>
<td>Full</td>
</tr>
<tr>
<td>Item Deletion Verification</td>
<td>✓</td>
<td>✓</td>
<td>Full</td>
<td>Full</td>
</tr>
<tr>
<td>Event Viewer</td>
<td>✓</td>
<td>✓</td>
<td>Full</td>
<td>Full</td>
</tr>
<tr>
<td>Sensor Monitoring</td>
<td>✓</td>
<td>✓</td>
<td>Full</td>
<td>Partial: Sensor data will be simulated</td>
</tr>
<tr>
<td>Contact Center</td>
<td>✓</td>
<td>✓</td>
<td>Full</td>
<td>Partial: Contact data will be simulated</td>
</tr>
<tr>
<td>Emergency Contacts</td>
<td>✓</td>
<td>✓</td>
<td>Full</td>
<td>Partial: Contact data will be simulated</td>
</tr>
<tr>
<td>Notifications</td>
<td>✓</td>
<td>✓</td>
<td>Full</td>
<td>Full</td>
</tr>
<tr>
<td>Remote Access</td>
<td>✓</td>
<td>✓</td>
<td>Full</td>
<td>Not implemented in the prototype</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DATA MANAGEMENT</th>
<th>Patient</th>
<th>Caregiver</th>
<th>RWP</th>
<th>Prototype</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Analytics</td>
<td>✓</td>
<td>✓</td>
<td>Full</td>
<td>Partial: Analytics will be based off simulated data</td>
</tr>
<tr>
<td>Reports</td>
<td>✓</td>
<td>✓</td>
<td>Full</td>
<td>Full</td>
</tr>
<tr>
<td>Item Tagging</td>
<td>✓</td>
<td>✓</td>
<td>Full</td>
<td>Partial: Geotag data will be simulated</td>
</tr>
<tr>
<td>Images</td>
<td>✓</td>
<td>✓</td>
<td>Full</td>
<td>Full</td>
</tr>
</tbody>
</table>

The prototype development for the application involves a mix of full and partial implementations of various features. Full implementation includes user account management, encompassing account creation, profile management, and login capabilities. Additionally, several user interface features are fully implemented, such as task management, calendar functions, addition and deletion of commonly lost items, and in-app notifications. However, some features requiring external data sources are only partially implemented, focusing on showcasing in-app functionality rather than relying on real-world data. For instance, item location data and contact information are simulated, but the emergency contact process will be fully implemented for practical use.
Remote access is omitted from the prototype as it doesn't directly address the primary concerns of dementia patients, being considered more of an optional feature for future consideration. Regarding data management, some aspects are fully implemented. User reports and item tag data will be simulated, but formatted reports and images used to track items are fully implemented, illustrating how caregivers can benefit from patient activity data to tailor their care plans.

4.3 Prototype Development Challenges

In the process of prototype development, several critical challenges must be addressed. These challenges include resource limitations, such as constrained time and labor resources, which can impact the development timeline. Technical feasibility may pose implementation challenges for specific features, potentially requiring creative solutions. Hardware and software compatibility issues may arise, particularly when dealing with diverse hardware and software configurations. Testing and bug fixing are imperative to identify and resolve issues during development, ensuring the prototype's correctness. Scalability is a fundamental consideration to ensure the prototype effectively handles real-world scenarios. Additionally, selecting the most suitable prototyping tools and platforms is essential to facilitate the development process effectively. Navigating these challenges is essential for the successful creation of our prototype.

[ This space left intentionally blank. ]
5 Glossary

**Amazon Web Services (AWS):** A cloud computing platform that provides a variety of services including compute, storage, databases, analytics, machine learning, networking, mobile, developer tools, security, and enterprise applications.

**Android:** An open-source mobile operating system based on the Linux kernel and developed by Google.

**Application Programming Interface (API):** A set of protocols, routines, and tools for building software applications that specify how software components should interact with each other.

**Application Server:** A software framework that provides an environment for running applications.

**Authentication:** The process of verifying the identity of a user.

**Biometric:** unique physical or behavioral characteristics of an individual to identify them.

**Bluetooth Low Energy (BLE):** a wireless communication technology used for short-range communication between devices.

**Caregiver/Family Member:** Any person that is able to provide assistance with managing a dementia patient’s symptoms. Their duties consist of transporting patients to and from their commitments, maintaining a patient’s daily routine, and reminding them to take their medications at regular intervals.

**Continuous Integration (CI):** A software development practice that involves frequently integrating code changes from multiple developers into a shared repository, verifying that the changes do not break the build and that the software continues to function correctly.
**Containerization:** A method of packaging and deploying software applications with all their dependencies into a single unit, called a container, which can run reliably and consistently across different computing environments.

**Database Server:** A computer program or software application that provides database services to other computer programs or clients.

**Docker:** A software platform that allows its users to build, test, and deploy applications in standardized executables resembling containers.

**GeoTag:** A physical tag that emits a Bluetooth signal to assist devices in determining their location.

**GitHub:** An open-source repository service that allows its users to work on a single project simultaneously.

**Google Maps Geolocation API:** A service provided by Google that allows developers to determine the location of a device using Wi-Fi or mobile network signals.

**iOS:** A mobile operating system developed by Apple Inc based on the Unix operating system.

**Issue Tracking:** The process of managing and resolving software issues, bugs, and feature requests.

**JavaScript:** Scripting programming language that creates dynamic web page content and mobile applications.

**JavaScript Object Notation (JSON):** A lightweight data interchange format that is easy for humans to read and write and easy for machines to parse and generate.

**Jest:** Test-runner for JavaScript applications that supports a JavaScript library for creating, running, and structuring tests.

**MongoDB:** Non-relational document database that provides support for non-relational querying.
**Node.js:** An open-source, cross-platform JavaScript runtime environment that enables the execution of server-side JavaScript code.

**Non-relational Database:** A type of database that doesn't rely on the traditional structure of tables, columns, and rows found in relational databases. Instead, they are designed to handle large and complex sets of unstructured, semi-structured, or structured data.

**Patient:** Any individual suffering from mild to moderate dementia. Their symptoms typically comprise of occasional disorientation, difficulties with making decisions, and short-term memory loss.

**PostgreSQL:** Free and open-source database management system that supports relational (SQL) and non-relational (JSON) querying.

**Programming Language:** A formal language used to communicate instructions to a computer or other machine.

**React:** An open-source JavaScript library that is used to build user interfaces for web and mobile applications.

**Relational Database:** A type of database that stores data in a structured format, using rows and columns to represent data entities and attributes.

**Relational Database Management System (RDBMS):** A software system that enables users to create, maintain, and manipulate relational databases.

**Repository:** A central location where digital files, usually in the form of software source code, are stored and managed.

**Structured Query Language (SQL):** A programming language used to manage and manipulate data in a relational database management system (RDBMS).
Testing Framework: A set of guidelines, standards, and tools that software developers use to create and run automated tests for their code.

Version Control: A system that tracks and manages changes to software code, documentation, and other files over time.
6 References


References


