Bilkent University

Department of Computer Engineering

Senior Design Project

Willow: Graph-Based Browsing

High-Level Design Report

Efe Dağdemir, Tuana Türkmen, Sezin Zeydan, Can Cebeci, Cem Cebeci

Supervisor: Uğur Doğrusöz
Jury Members: Çiğdem Gündüz Demir, Can Alkan

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Senior Design Project course CS491/2.

Table of Contents

1. Introduction

1.1 Purpose of the system

Millions of people are accessing the World Wide Web every single minute and web browsers are helping them do it. Although reaching the web by a browser is simple and easy, navigating between numerous tabs and managing different browsing sessions are not. One can easily get lost in between abundant tabs and lose track of the tasks in hand. As the number of tabs get larger, the amount of time lost due to confusion increases. Furthermore, backtracking on multiple tabs gets troublesome on the current model offered by the browsers. The features offered by web browsers for such tasks are limited and outdated. For instance, the linear structure of the tabs becomes hard to use as the size of the tabs gets smaller and identifying the desired tab between many others becomes burdensome with each new tab. Additionally, the concept of not being able to access a collection of tabs from previous browsing experiences is a flexibility limitation for the users. Willow's purpose is to be a solution to the problems mentioned above.

Willow is a Chrome Extension that could help to ease the process of browsing via understandable and intuitive visual navigation, filtering techniques and session management features. Willow aims to make users save time and feel comfortable during browsing.

1.2 Design goals

Modifiability:

Willow should be easily modifiable when changes arrive. The underlying system should not be complex, the size of the components must not be too large, the components must be named appropriately and it should be conveniently documented in order to make it easily understandable for future development. The system structure should have minimized coupling and maximum cohesion to prevent a change from causing other rippling changes in the system. The amount of time it takes to deploy an updated version corresponding to the change should not be long.

Efficiency:

For Willow, timing is crucial because synchronizing with the browser activity of the user is one of its most important features. So Willow should be implemented in an efficient way to be able to update itself continuously.

User-friendliness:

Willow's most basic purpose is to ease the browsing process of the users. Thus it is important for users to have a smooth experience while using Willow. We aim to avoid overwhelming the users by providing an understandable and simple interface. At the same time, a user manual which informs the reader about the features and usage of Willow will be provided.

Adaptability:

As stated many times, Willow will be a Chrome extension. The provided APIs from Chrome will be used. Thus, it is crucial for the system to be adaptable to the changes that might occur within these APIs so that Willow can continue functioning as time passes and changes occur.

1.3 Definitions, acronyms, and abbreviations

Session: Sessions are continuous instances of browsing activity. A new session starts every time the user launches the Chrome app. The session is terminated upon exit. Sessions can be saved and restored, allowing them to last across multiple runs of Chrome.
Session Graph: A visual graph structure that shows the relationships between nodes. The session graph contains an edge u → v if and only if the first instance of access to the website associated with node v was through a link contained in the website associated with node u.
Node: A vertex on the session graph which represents a web page visited within the session.
Open Node: A node whose associated page is currently open in a tab.
Closed Node: A node whose associated page was visited within the session and is not currently open in a tab. The tab that once contained the web page need does not have to be closed, it may simply be navigated to another web page.
Active Tab: The tab whose content is currently displayed. At any time, the browser contains exactly one active tab and any number (possibly zero) of inactive tabs.
Active Node: The node that is associated with the web page contained in the active tab. Trivially, there is exactly one active node at all times and the active node is an open node.
Willow Overlay / Graph Overlay: The hideable and extendable panel that is added to Chrome's UI when Willow is installed. The overlay contains the session graph and is the main medium of interaction with Willow.

1.4 Overview

This report is the high-level design report for Willow which will convey the current system, the proposed software architecture and subsystem services. This report also contains information about various engineering and ethical considerations and our teamwork.

2. Current software architecture

The current system is Google Chrome's tab management system [1], which is not very flexible for users that work with many open tabs.
Chrome's tabs have some properties like *active, id, incognito, selected, sessionId, windowId,* etc. that help the managing of tabs.
Right now, Google Chrome offers tab management through features such as:

- All open tabs on a Chrome window appear side to side on top of the window based on the order in which they were opened.
- If a user wants to, they can change the order of the open tabs that are displayed.
- Users can move tabs between different Chrome windows.
- Users can hover their mouse on a tab and a tooltip containing a brief description of the web page such as its name and title is shown.
- Users can retrieve closed tabs from their history.
- Users can edit their history.
- Users can bookmark the tabs they deem important and have quick access to those tabs through Bookmarks. Users can also edit their bookmarks.

- When users shut down the chrome app and then launch Chrome again, all web pages from the last browsing session can be restored.
- The relationships that exist among tabs is not and can not be indicated, except by the primitive feature of tab groups, which can be constructed manually to underline chosen tabs with the same color and keep them adjacent in the order the tabs are displayed in.

Although the above features are useful, they are not enough for a user that surfs the web with a large number of open tabs.
Chrome offers useful APIs for developers who want to work with Chrome's features which can be very useful for the Willow Extension. Some of them are chrome.tabs, chrome.history, chrome.sessions, chrome.bookmarks, chrome.commands, chrome.extension [2].

- chrome.tabs provides an interaction with the browser's tab system. Tabs can be created, modified, rearranged via this API.
- chrome.history provides communication with the browser's visited web pages records. URLs in the history can be removed, queried or added via this API.
- chrome.sessions provides querying and restoring of tabs and windows from a browsing session.
- chrome.bookmarks is for creating, modifying and organizing bookmarks.
- chrome.commands can be used for adding keyboard shortcuts to extensions.
- chrome.storage provides ways to store, monitor and retrieve the changes of users' data.

3. Proposed software architecture

3.1 Overview

Below is the explanation of the proposed software architecture for Willow.

3.2 Subsystem decomposition

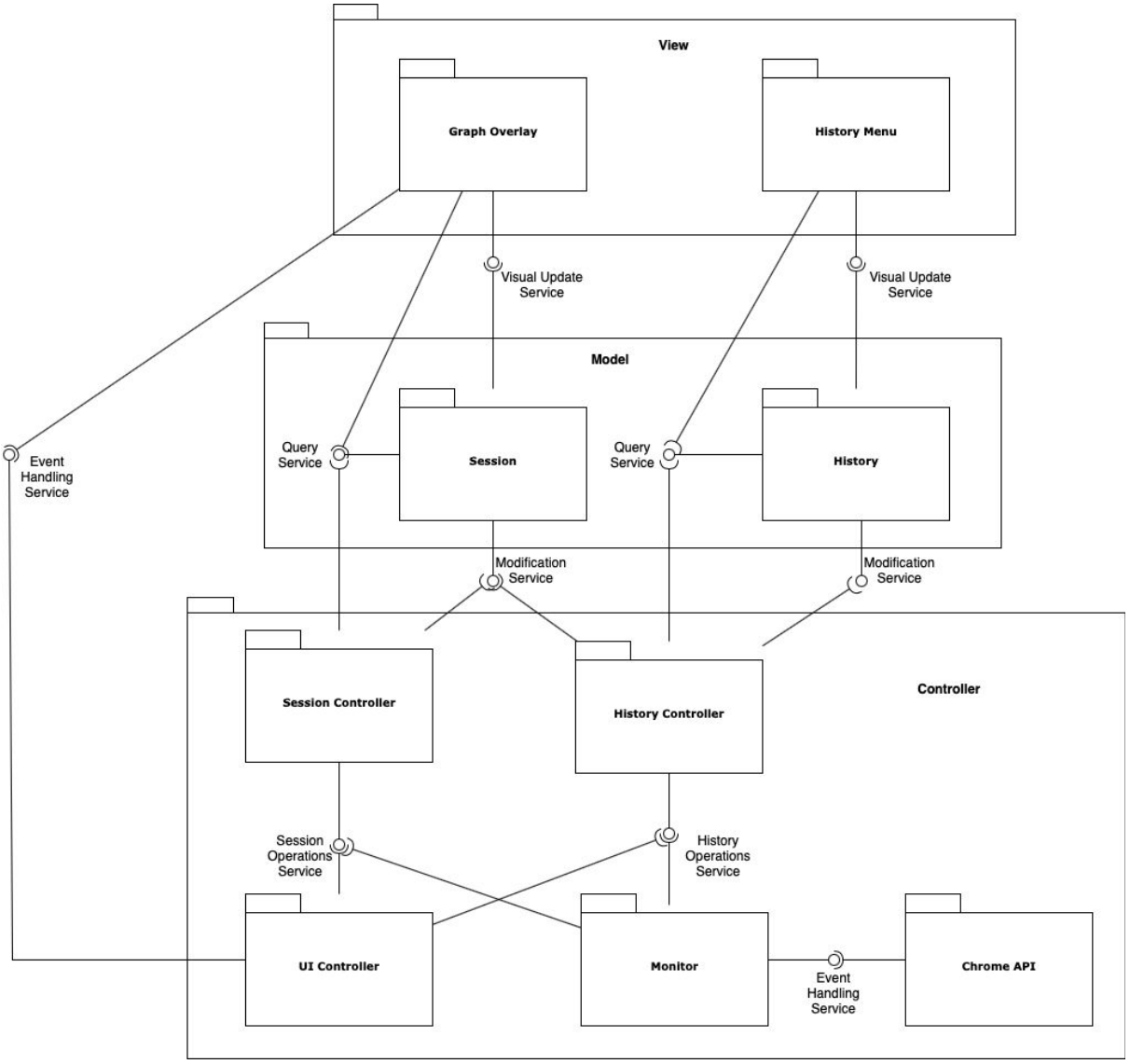

Figure 1: Subsystem Decomposition

3.3 Hardware/software mapping

Willow is a browser extension that is specific to the Google Chrome web browser. Which means that Willow is capable of running on any version of Google Chrome that supports extensions, regardless of the platform. The subsystems of Willow will be realized entirely by software.

Subsystems that deal with tabs, browsing data, session management and storage will make use of the resources provided by numerous special-purpose APIs of Google Chrome. Additionally, 3rd party libraries such as Cytoscape.js will be used to visualize the graph. The

development of the extension will be done with Javascript. The interfaces will be implemented with HTML and CSS in addition to Javascript.
As the data will be stored locally via the storage resources provided by the Chrome API, a database will not be needed.

3.4 Persistent data management

Persistent data is essential for session saving and restoration functionalities of Willow. The data that must be persistently stored includes the sessions of the user, the browsing history of the session, the hierarchical interconnectivity of the nodes in the session graph and the customized visualization information of the session graph. As the mentioned data is specific to the users and can be changed frequently, it is best to store it locally. Additionally, as concurrency control is not needed for a single user and a heavy relational scheme between the persistently stored objects is non-existent, there is no need for a database management system. In accordance with these, the data will be stored with a text-based, lightweight, structured data format such as JSON. Therefore, Google Chrome's storage API will be used to meet our needs of persistently managing data as it allows us to store, retrieve and track changes to local user data in text-based format.

3.5 Access control and security

In Willow there is only one type of user thus all users have the same accesses and permissions in the application. Each user has their own data stored locally on their computer. The users can access their current session, history of saved sessions labeled either by date or by the name the user gave to the session.

The application is a Chrome extension, each user must be logged in to their Chrome accounts before installing the extension. After installing Willow users will be prompted to give Willow access to read their Chrome browsing history. Unless users allow access to their browsing history Willow will not be launched. The users will not have access to other users name, profile or browsing history. In Wilow there is no interaction between users. Also for Willow to save their data locally users should give storage access to Willow.

Another security issue is that after installing Willow, if the user wishes they can turn off Willow and browse freely without Willow tracking and organising their session. Willow guarantees that users' browsing will not be tracked until the user explicitly permits it.

3.6 Global software control

Global software control in Willow follows the event-driven control mechanism with a decentralized design. User actions control the behavior of the system. Without user inputs, the system is stationary. As the user provides inputs by opening tabs and visiting web pages, the session graph is updated in an event-driven manner. There are multiple control objects inside the system. The responsibilities of the control objects are divided in accordance with their functionalities. For example, the tab control mechanism and session saving/restoring mechanisms are responsibilities of different control objects. This way, the global software control structure of Willow fits better into object-oriented development.

3.7 Boundary conditions

From the point of view of the user Willow has three boundary conditions: Initialization, termination, failure.

Initialization

Assuming the user is logged into their Chrome account, after the user clicks the install button they see a prompt asking for their permission for Willow to access their browsing data and storage. Unless users allow this, Willow will not be installed. After the user allows necessary access for Willow to operate, Willow will be installed and the Willow logo will appear on the extension bar on users' Chrome window. When the installation process is done the user can launch Willow and start a session. From that point on Willow will track the user's browsing on Chrome and provide a visual diagram for their browsing session unless the user explicitly tells Willow not to.

Termination

Willow will be actively tracking and providing a visual graph of users' browsing session unless the user explicitly disables Willow. This means that when the user is browsing and Willow is not disabled, Willow will keep detecting events. Willow will stop working if the user clicks the disable button. Willow also terminates when Chrome terminates, making sure to save the current session in the history before doing so.

Failure

If Chrome terminates unexpectedly a failure will occur. In case of an unexpected shutdown Google Chrome tabs will retrieve the old session back however this is not possible for Willow. If an unexpected power loss occurs Willow does not save the current session so the session information will be lost. This may lead to synchronization problems with the Chrome history.

4. Subsystem services

4.1 Session Subsystem

Using the related Chrome APIs, stores the data about the active session such as the hierarchical graph structure, the web pages each node is related to and the tabs. Handles both volatile and persistent storage for the session.

Modification Service

add _node(node, parent)

Adds the given node as a child of the parent in the parameter.

remove_node(node)

Removes the given node from the session graph. It's children may be removed or transferred to its parent.

set_session_name(name)

Changes the current session's name.

change_session(session)
Changes the current session to a different session.

move_tab(tab, target_node)
Associates the tab with the target node, removing its previous association.

create_tab()
Creates a new tab with the default content.

remove_tab(tab)
Removes the given tab from the list of tabs.

get_active_tab()
Returns the information about the active tab.

set_position(node, position)
Changes the position of a node.

set_size(node, size)
Changes the size of a node.

set_color(node, color)
Changes the color of a node.

Query Service

get_root_node()
Returns the root node of the session graph.

get_children(node)
Returns all children of a node.

get_active_node()
Returns the node corresponding to the active tab.

get_session_name()
Returns the current session's name.

get_tabs()
Returns all tabs in the current session.

get_active_tab()
Returns the active tab

Subscription Service

subscribe_to_node(subscriber, node)
Sets up a subscriber to be notified when changes happen to a node.

4.2 History Subsystem

Using the related Chrome APIs, stores the data about the history, organized in sessions. Handles both volatile and persistent storage.

Modification Service

create_session()
Creates a new empty session.

set_session_name(session, name)
Changes the name of a given session.

set_session_access_date(session,date)
Changes the session last access date.

remove_session(session)
Removes the session from the list of sessions in the history.

Query Service

subscribe_to_session(session)
Sets up a subscriber to be notified when changes happen to a session.

get_sessions()
Returns all previous and current sessions.

4.3 Session Controller Subsystem

Controls the interaction between the user and the session model. Makes the necessary changes to the session model when the user performs an action. The operations described here are similar to the operations of the session subsystem, the operations in here invoke the other ones after performing validity checks and business logic to alter the state of a session.

Session Operations Service

remove_node(node)
Removes the node from the session, requests the actual change from the session subsystem.

add_node(node, parent)
Adds the node as a child of parent, requests the actual change from the session subsystem.

resize_node(node, size)
Changes the size of a node.

move_node(node, position)

Changes the position of a node.

change_node_color(node, color)
Changes the color of a node.

switch_active_tab(tab)
Sets the given tab as the active tab, the previous active tab will not be active after this call.

change_tab_content(tab, content)
Changes the content of a tab to a node.

close_tab(tab)
Closes the given tab.

create_tab()
Creates a new tab with the default node as its content.

change_active_session(session)
Replaces the current active session with another session.

4.4 History Controller Subsystem

Controls the interaction between the user and the history model. Makes the necessary changes to the history model when the user performs an action. The operations described here are similar to the operations of the history, the operations in here invoke the operations in the history subsystem after performing validity checks and business logic to alter the state of a session.

History Operations Service

start_new_session()
Starts a new session and registers it to the history.

remove_session(session)
Removes the session from the list of all sessions.

rename_session(session, name)
Changes the name of a session.

update_access_date(session)
Changes the session's last access date to the current date.

4.5 Monitor Subsystem

In contrast to the UI Controller Subsystem's handling of the user's interaction with the explicit UI elements of Willow, the Monitor Subsystem tracks and handles user activity in the browser that the user does not explicitly communicate to Willow. Tracks when links are

clicked, new web pages are visited and tabs are re-navigated, created or closed. Monitor subsystem makes sure that Willow's model properly reflects the user's browsing session and their interaction with Chrome.

Since this subsystem serves the purpose of event detection, its functions are not invoked explicitly by other subsystems. Instead, it adds various listeners to Chrome in order to detect relevant user activity and uses the operations of its client subsystems to notify them of events.

4.6 UI Controller Subsystem

Listens to the UI elements that are added to Chrome by Willow and forwards any explicit user interaction to the related subsystem.

Event Handling Service

button_pressed(event)
Called when a button that is a part of Willow's UI is pressed. Depending on the button, forwards the event to the appropriate subsystem.

mouse_dragged(event)
Called when the mouse is dragged over an element that is a part of Willow's UI. Depending on the element, forwards the event to the appropriate subsystem.

mouse_scrolled(event)
Called when the mouse wheel is scrolled over an element that is a part of Willow's UI. Depending on the element, forwards the event to the appropriate subsystem.

node_clicked(event)
Called when a node on the visual representation of the session graph is clicked. A right click will be forwarded to the Graph Overlay Subsystem in order to open a context menu. A left click will cause the web page related to the clicked node to be opened in the active tab or activate a tab containing the related page if there is one. The Session Controller Subsystem will also be notified of the effects.

edge_clicked(event)
Called when an edge on the visual representation of the session graph is clicked. A right click will be forwarded to the Graph Overlay Subsystem in order to open a context menu. A left click will have no effect

4.7 History Menu Subsystem

The view subsystem that is responsible for the history menu.

Visual Update Service

draw_history_menu()
Creates a menu displaying the Willow history, organized as a collection of browsing sessions.

draw_session_preview(session)
Creates a visual preview of a previous browsing session.

4.8 Graph Overlay Subsystem

The view subsystem that is responsible for displaying the session graph.

Visual Update Service

draw_node(web page, size, position, color)
Inserts a node into the session graph with the input visual parameters

draw_edge(parent, child)
Inserts a directed edge between two nodes in the session graph.

erase_node(node)
Removes a node from the session graph.

erase_edge(edge)
Removes an edge from the session graph.

draw_tab_marker(node)
Called for nodes that are related to web pages open in a tab. Adds a visual indicator to the node in the graph.

erase_tab_marker(node)
If the node has a tab marker, removes the marker.

activate_tab_marker(node)
If the node has a tab marker, applies a distinction to the marker, indicating that the tab containing the web page is active.

deactivate_tab_marker(node)
Reverts an active tab marker to the regular tab marker.

change_zoom_level(zoom_level)
Changes the zoom level of the view.

change_camera_pos(camera_pos)
Causes the view to pan and scroll.

move_node(position)
Changes the position of a node in the view.

resize_node(size)
Changes the size of a node in the view.

change_node_color(size)
Changes the color of a node in the view.

draw_context_menu(menu_kind)
Draws a context menu next to the mouse. The menu displays various operations, dependent on menu_kind, that can be invoked.

5. Consideration of Various Factors in Engineering Design

Willow is a tool designed to enhance people's internet browsing experience. Hence, it does not affect and is not affected by public health, public safety and public welfare in any way at all. Social factors are also irrelevant for the design of Willow.

Global Factors:
Willow is imagined as a software to be used by people all around the world. However, people speak many different languages all around the globe. Considering that most people that browse the internet speak English, having the interface in English will allow people from different countries to access the software.

Cultural Factors:
Different cultures have different understandings of privacy. The European Union has very strict rules about data protection. On the other hand, the Chinese are used to a much more invasive mode of surveillance. In designing Willow, the privacy boundaries of different cultures need to be respected.

Table 1: Factors

	Effect Level	Effect
Public Health	0	-
Public Safety	0	-
Public Welfare	0	-
Global Factors	4	Interface should be English
Cultural Factors	6	Privacy and Data Protection
Social Factors	0	-

6. Teamwork Details

6.1 Contributing and functioning effectively on the team

As the Willow team we believe that the road to proper teamwork is through healthy communication. To do that we meet at least once a week and discuss our progress that week. In these meetings new tasks are assigned to every group member in a fair manner and important decisions about design, implementation or feasibility are made. We do every task on a schedule. We give proper deadlines for each task and try to keep up with those deadlines and this ensures that we keep proper time tables and never get behind schedule. Also, all of our team members have previously worked on various course projects together. We believe that our familiarity with each other's strengths, weaknesses, working habits and communication styles will help us maintain a respectful and comfortable work environment and keep us on track.

Another caution we take to ensure proper and effective teamwork is that we do not do every task individually. For example if it is a very big task we divide the task in parts and we also divide the group and then start doing it each smaller task for each group. When it's time to combine the mini tasks we come together to review and discuss the tasks done by each group. This approach is advantageous because it is very efficient and also gives everyone in the group to review the task as a whole in the end.

6.2 Helping creating a collaborative and inclusive environment

Our primary aim as Team Willow is to follow the software development life cycle properly. In order to achieve this we will analyze and design our project comprehensively and we plan on implementing a prototype based on our analysis and design before the end of the fall semester. Creating a prototype is important for us because we want to receive qualitative and quantitative feedback from users as soon as possible. This will allow us to test our product on the field and make continuous improvements in the necessary areas. After making sure our product is shaped to meet the initially planned functionalities and the demands of the users, we will move on to implementing our extended features. We plan on testing our extended features with users and receiving feedback in the same way with the prototype of the core product.

6.3 Taking lead role and sharing leadership on the team

According to our plan we have 9 work packages: Analysis Report, High-Level Design Report, First Prototype, Marketing, Low-Level Design Report, Second Prototype, Marketing, Final Implementation Changes, Final Report. And we had one previous work package for Requirements Specification.

Table 2: List of work packages

WP#	Work package title	Leader	Members involved
WP0	Requirements Specification	Sezin Zeydan	All members
WP1	Analysis Report	Tuana Türkmen	All members
WP2	High-Level Design Report	Cem Cebeci	All members
WP3	First Prototype	Efe Dağdemir	All members
WP4	Marketing	Can Cebeci	Sezin Zeydan, Efe Dağdemir
WP5	Low-Level Design Report	Sezin Zeydan	All members
WP6	Second Prototype	Can Cebeci	All members
WP7	Marketing	Tuana Türkmen	Cem Cebeci
WP8	Final Implementation Changes	Cem Cebeci	All members
WP9	Final Report	Efe Dağdemir	All members

Table 3: Explanation of the work packages

WP 0: *Requirements Specification*

Start date: *Sep 16, 2020* **End date:** *Oct 12, 2020*			
Leader:	*Sezin Zeydan*	**Members involved:**	*All members (Tuana Türkmen, Efe Dağdemir, Cem Cebeci, Can Cebeci)*
Objectives: *Name and briefly describe the project. Identify initial project requirements, constraints and professional & ethical responsibilities.*			
Tasks: ***Task 0.1 Functional Requirements*** **:** Identifying the functional requirements. ***Task 0.2 Constraints*****:** *Determining the constraints.*			
Deliverables ***D0.1:*** Project Specification Report			
WP 1: *Analysis Report*			
Start date: *Oct 12, 2020* **End date:** *Nov, 21 2020*			
Leader:	*Tuana Türkmen*	**Members involved:**	*All members (Efe Dağdemir, Sezin Zeydan, Cem Cebeci, Can Cebeci)*
Objectives: *Having a comprehensive analysis of the project so that the design and implementation of the project goes as smoothly as possible.*			
Tasks: ***Task 1.1 Research about the current system*****:** *The purpose is to know the current system. This is important for the design and implementation of the project because we would know*			

what is already available, what should be extra implemented and what can be changed in the current system.

Task 1.2 Analyzing the proposed system: *The purpose is to decide on the functionalities and the models of the system.*

Task 1.3 Considering social conditions: *The purpose is to decide on risks, project plan, responsibilities, etc.*

Deliverables

D1.1: *Analysis Report*

WP2: *High-Level Design Report*

Start date: *Nov, 21 2020* **End date:** Dec 21, 2020

Leader:	*Cem Cebeci*	**Members involved:**	*All members (Efe Dağdemir, Sezin Zeydan, Tuana Türkmen, Can Cebeci)*

Objectives: *Having a comprehensive design of the project so that the implementation of the project goes as smoothly as possible.*

Tasks:

Task 2.1 Architecture: *The purpose is to decide on which architecture we will use while implementing our project.*

Task 2.2 System Models**:** *The purpose is to have more precise system models that we will use during our implementation.*

Deliverables

D2.1: *High-Level Design Report*

WP3: *First Prototype*

Start date: Dec 21, 2020 **End date:** Before the end of fall semester			
Leader:	*Efe Dağdemir*	**Members involved:**	*All members (Sezin Zeydan, Tuana Türkmen, Can Cebeci, Cem Cebeci)*
Objectives: *Implementing a presentable prototype with basic functionality.*			
Tasks: ***Task 3.1 Implementation:*** *Implementing the project so that it has basic functionality and can be tested by presenting to ısers.*			
Deliverables ***D3.1:*** *A basic functioning prototype*			
WP4: *Marketing*			
Start date: After implementation of the prototype **End date:** Before the demo on the fall semester			
Leader:	*Can Cebeci*	**Members involved:**	*Sezin Zeydan, Efe Dağdemir*
Objectives: *Marketing strategies for populating the product between users.*			
Task 4.1 Social Media: *Using social media platforms to introduce users with the product.*			

WP5: *Low-Level Design Report*			
Start date: *1st week of the spring semester* **End date:** *3rd week of the spring semester*			
Leader:	*Sezin Zeydan*	**Members involved:**	*All members (Efe Dağdemir, Tuana Türkmen, Can Cebeci, Cem Cebeci)*
Objectives: *Having a comprehensive low-level design of the project so that the implementation of the project goes as smoothly as possible.*			
Tasks: ***Task 5.1 Architecture***: *The purpose is to decide on which architecture we will use while implementing our project.* ***Task 5.2 System Models*****:** *The purpose is to have more precise system models that we will use during our implementation.*			
Deliverables ***D5.1:*** *Low-Level Design Report*			
WP6: *Second Prototype*			
Start date: 3rd week of the spring semester **End date:** Before the Final Report deadline			
Leader:	*Can Cebeci*	**Members involved:**	*All members (Sezin Zeydan, Tuana Türkmen, Efe Dağdemir, Cem Cebeci)*

Objectives: Implementing a second prototype according to the user feedback we received which optimistically will have more features.

Tasks:

Task 6.1 Changes: *The purpose is to change the first prototype according to the feedback received.*

Task 6.1 Extended Features: *The purpose is to add the extended features to the implementation.*

Deliverables

D6.1: Second Prototype

WP7: *Marketing*

Start date: After implementation of the second prototype **End date:** Before the final changes

Leader:	*Tuana Türkmen*	**Members involved:**	*Cem Cebeci*

Objectives: *Marketing strategies for populating the product between users.*

Task 7.1 Social Media: *Using social media platforms to introduce users with the product.*

WP8: Final Implementation Changes

Start date: After implementation and marketing of the second prototype **End date:** Before CSFair			
Leader:	*Cem Cebeci*	**Members involved:**	*All members (Efe Dağdemir, Sezin Zeydan, Tuana Türkmen, Can Cebeci)*
Objectives: *Having the final implementation of the project.*			
Task 8.1 Finalizing: *Finalizing the project implementation according to the latest user feedback and having an extensive testing procedure.*			
Deliverables ***D8.1:*** *Final Product*			
WP9: *Final Report*			
Leader:	*Efe Dağdemir*	**Members involved:**	*All members (Sezin Zeydan, Tuana Türkmen, Can Cebeci, Cem Cebeci)*
Objectives: *Provide a comprehensive final report of the project that gives complete information about the system.*			
Tasks: ***Task 9.1 Architecture***: Final architecture of the system. ***Task 9.2 Maintenance Plan:*** Provide a maintenance plan for the system. ***Task 9.3 Social Impact:*** Analyse social impact of the system.			

Deliverables

D9.1: *Final Report*

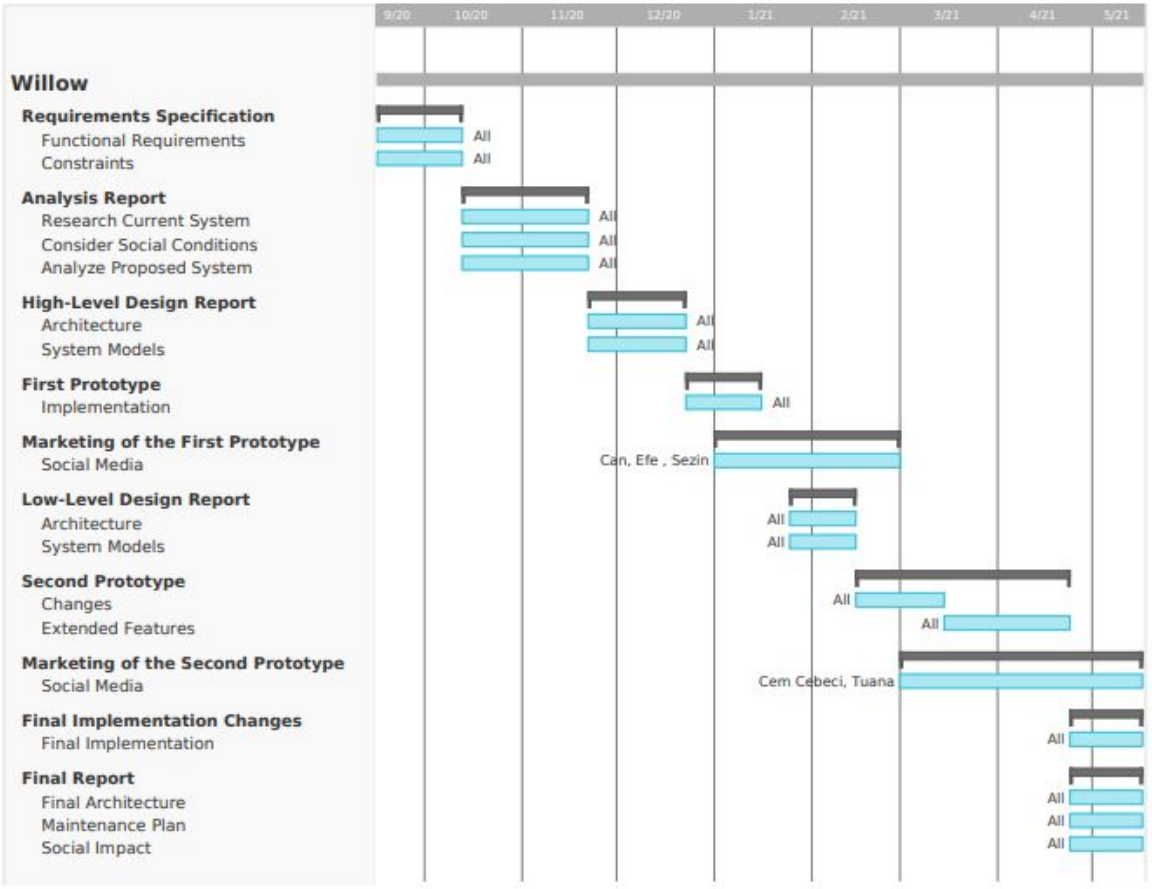

Figure 2: Gantt chart of the project plan

7. References

[1] “Keep tabs on your tabs in Google Chrome ” Available: https://blog.google/products/chrome/manage-tabs-with-google-chrome/ [Accessed: 20-Nov-2020].

[2] “API Reference,” *Chrome Developers*. [Online]. Available: https://developer.chrome.com/docs/extensions/reference/. [Accessed: 24-Dec-2020].