Team Bailey Project Proposal

“PAL” AI Assistant for Gaming

## Team Members

Eren Bikmaz

Lehi Briggs

Morgan Anderson

Bailey Goins

# 

# Project description

There are several AI assistants on the market, but they are all aimed at a generic audience and thus can only do generic tasks. We intend to create an assistant for the PC gamer demographic which is globally increasing faster than soccer in viewers and participants. PC gamers often find themselves in the middle of fast-paced action that grinds to a halt whenever they need to perform some other task on their desktop. While Microsoft’s Cortana can perform simple actions while playing a game, it’s functionality is limited in regards to what a PC gamer needs from an assistant. Mobile assistants such as Google Assistant, Apple’s Siri, and Samsung’s Bixby provide a more extensive set of commands but require PC gamers to look away from their game to interact with these assistants on their mobile devices. Our goal with PAL is to build a hands-free experience for PC gamers that allows them to perform tasks while their full attention remains on their game.

## Product capabilities (in order of priority)

1. Controlling windows in other screens and/or a non-invasive in-game overlay mouse-free using voice commands.
   1. Twitch
      1. Start/stop watching streams on second monitors/pop-out window
      2. Start streaming from specific monitors/applications
      3. Ask how many subscribers a user has
      4. Subscribe/unsubscribe to/from users
   2. Spotify
      1. Playing/pausing/skipping songs or playlists
      2. Searching for songs or playlists
      3. Skipping songs
   3. Browser Windows
      1. Searching for/playing YouTube videos
      2. Queuing shows on Crunchyroll
      3. Opening/displaying web pages, performing searches
   4. IO management
      1. Changing audio input/output devices
      2. Volume control
2. Social Integration with Discord, Steam, etc.
   1. Get friends status
      1. See if your friend is online
      2. See what game they are playing
      3. See if they are streaming or not
   2. Discord
      1. Start streaming in group chat
      2. Mute voice channel
      3. Opening/reading/sending messages,
      4. making/accepting/rejecting calls
   3. Steam
      1. Messaging users and friends
      2. Game activity
      3. Searching for active friends
      4. Asking what games friends are playing
      5. Making/accepting game invites

## Milestones

1. Voice recognition - our Assistant will be able to recognize most commands spoken by the user
2. Voice responses - after every command our Assistant will relay to the user either a confirmation response or an error response telling the user what went wrong
3. Task integrations - our Assistant must be able to accomplish these basic tasks:
   1. Play and pause Youtube videos that are running
   2. Send a message through discord
   3. Open Twitch stream of a specific streamer.
   4. Open a window on a specific desktop screen
   5. Check the activity status of friends on steam
   6. Start playing a specific anime episode
4. GUI overlay that rests on top of applications in a nonintrusive manner

## Compatibility

Hardware - PAL will be built to be compatible with Windows 10 desktop PCs and newer, while PAL might later be extended to be compatible with other devices and operating systems this will not be a core feature.

Software - PAL will be compatible with Discord, Google Chrome, OBS, and Twitch. Future software integrations are possible however these are the main systems PAL will leverage

## 

## User Stories

|  |  |  |
| --- | --- | --- |
| As a... | I want to... | So that... |
| Gamer | Swap windows on other screens while gaming with my voice | Check/change windows quickly while in-game without interrupting gameplay |
| Gamer | Perform voice searches while gaming and have them displayed over my game | I can perform searches and read the results without interrupting gameplay |
| Streamer | Start streaming using voice commands | I can start a stream in the middle of a game without interrupting gameplay |
| Gamer | Start watching a stream/video on a second monitor or popup window using voice commands | I can watch videos/streams in a small window over my game without interrupting gameplay |
| Streamer | Open/read/send Discord messages on a second screen while streaming using voice commands | I can open/read/send messages discreetly |
| Gamer / Streamer | Play songs/playlists on Spotify using voice commands | I can change songs/playlists without interrupting gameplay |
| Gamer | Send/receive/accept game invites using voice commands | I can accept send/receive/accept game invites while washing dishes |
| Youtube User/ Anime Fan | Queue Youtube/Crunchyroll videos on a second monitor or popup window using voice commands | I can make changes to my queue without interrupting gameplay |
| Gamer / Streamer | Integrate my social networking platforms | I can ask PAL how many friends/followers/subscribers I have |
| Gamer | Control a second monitor using voice commands | I can do things on my desktop without messing up my game that doesn't have a borderless window mode |
| User | Be told when my command is not recognized by the assistant | I can know when I need to try again |
| User | Toggle the assistants listening on and off | I can prevent the assistant from being annoying |
| Admin | Be able to push a new command or set of commands to our users | When we find new functionalities for our users we can share it with them |

## 

## 

## Product Backlog

|  |  |
| --- | --- |
| Priority | Item |
| 1 | Create base GUI |
| 1 | Create the command recognition engine |
| 1 | Design command execution strategy |
| 1 | Build command to open Youtube |
| 2 | Create overlay GUI |
| 2 | Build volume control strategy |
| 2 | Add pause and play functionality |
| 2 | Render Youtube video on overlay |
| 3 | Refine speech recognition |
| 3 | Create contacts storage? |
| 4 | Discord integration |
| 4 | Send a message in Discord |
| 4 | Read the latest message in Discord |
| 5 | Twitch integration |
| 5 | Open stream |
| 5 | Pause/Play stream |
| 5 | Volume control |
| 6 | Spotify integration |
| 6 | Find song |
| 6 | Play/pause song |
| 6 | Shuffle |

# 

# 

# Scope and Risk

1. Time - Our first and biggest risk factor is the time and scope of the project.
2. Implementation - Using existing AI assistants may restrict functionality, building an assistant from existing libraries may prove to be far more difficult than we first imagined.
3. Reinventing the wheel - We may end up with an assistant that is simply a dumbed-down version of ones that may exist because we were not able to achieve the functionality we had hoped for.
4. Interfacing with applications is far easier than interfacing with web pages.
5. There is a “great leap” functionality-wise before the assistant can hear and respond to requests. This “feature gap” also exists for optional features such as integrating other AIs

## Assumptions

1. APIs exist to easily implement voice recognition
2. APIs exist to easily automate desktop tasks (opening programs, doing particular tasks)
3. It’s possible to interface with web browsers and simulate scrolling, clicking, and keyboard input, and this accessibility is somewhat uniform between browsers
4. Detecting and integrating with other platforms (Discord, Steam, etc..) will be compatible with our system
5. APIs for other assistants will be flexible enough to allow for integration both ways

## Constraints

1. Fixed due date: June 4th, 2021
2. Primary focus on Windows Platform, UNIX may be incompatible depending on the choice of API
3. Focus on using existing libraries over developing our own

## Stretch Goals

1. Integration with other assistants, one-way or both ways
   1. Sending commands to/from Google Assistant/Cortana/Alexa
   2. Communication with smart home devices
2. Add functionality to allow the user to create their custom commands (like macros)
   1. Run certain programs
   2. Execute predefined hotkeys

## Potential Frameworks

AI Systems:

* Google Cloud Speech
* An incredibly verbose toolset that offers speech-to-text and text-to-speech libraries
* Works with multiple languages and platforms
* Connecting to cloud APIs is cumbersome and requires substantially more overhead
* Highly complex, and requires a deeper knowledge of speech recognition and synthesis
* Microsoft Speech
* Easy to use, automates tasks involved with speech recognition and speech synthesis, does not require a deeper knowledge of speech recognition and synthesis
* Not nearly as verbose; it relies heavily on pre-established functions that are not extendable
* Restricted to Microsoft platforms.
* Pytorch
* Is an easily integratable library provides sophisticated speech recognition and synthesis
* Natural, human-like text-to-speech.
* Highly complex, and requires a deeper knowledge of speech recognition and synthesis
* Restricted to Python platforms
* Uses GPU rendering as opposed to CPU rendering

GUI / Overlay:

* UWP
* Windows Native, lots of support
* Everyone is familiar due to GUIs class
* Our app is only available through the Windows marketplace
* Limited to Windows 10
* Electron
* Open-source
* Overlay libraries publicly available
* Compatible with almost everything
* Javascript & HTML based
* Browser-based: poorer performance
* Overwolf
* Natively supports in-game overlays
* Lehi has met the devs in person
* “Walled Garden” like UWP; our app only available through their storefront
* Browser-based: poorer performance