|  |  |
| --- | --- |
| ARL Project  CS 448 Austin Bos, Aaron Saucedo, Ivan Nieto | Image result for army research laboratory |

SOW 000 for notetaking application requested by the Army Research Laboratory

|  |
| --- |
| Date |
| 02/08/2020 |

# Introduction

The purpose of this application is to allow its users to create and assign tasks through a GUI interface that will be saved on a local database. These tasks will have findings added to them that will allow the user to write their findings and upload images. The application will also have the ability to sync its local database with other copies of the application on different devices through a local network.

# Scope of work to be performed

|  |  |
| --- | --- |
|  | What we plan to accomplish in this project. |

For this application we will be developing a react based frontend, an SQLite database to store all the information for the application and a python backend that will handle the synchronization functionality between applications and the communication between the frontend and database. The application will be created using Electron to package all aspects of the application into a single executable. The sync functionality will be handled by the python backend and will use GRPC to send the information over the network. The backend will also handle the

* Frontend
  + Our frontend will be developed in HTML, CSS and JavaScript using React. The front end will consist of a number of pages that will allow the user to navigate the site and segment the different aspects of the application into a more simplistic interface. The main pages that we will develop will be an initial sign in page to assign a username for the user of the application, an overview page that will allow the user to see all their tasks for a specific project in one place, a task creating page used to create tasks and a task overview page that will allow the user to add findings to an individual task. More pages will be added as they are needed.
* Backend
  + The backend will be build using python and will be accesses from the frontend using Electrons build in systems. We will create multiple python scripts to insert and query the different tables in the database. The backend will also handle any synchronization between application over the network by using GRCP to handle the network configuration.
* Database
  + We will be using SQLite for our database. The database will have multiple tables including a table for users, projects, tasks and findings as well as any other tables that may be required.
* Sync Functionality
  + The sync functionality will be handled by the backend python scripts using GRPC.
    - The user will have an option in the application to connect with another computer and choose to sync with that machine.
    - The team lead will be able to sync from all machines on the network.
    - If we can (time/skill permitting), multiple users will be able to specify machines to stay in sync with, allowing real time updates of shared notes.
* Data Export

# Period of performance

|  |  |
| --- | --- |
|  | Timeframe available for development |

January 22, 2020 - May 8. 2020

# Work requirements

|  |  |
| --- | --- |
|  | Required tasks for the project |

* PHASE 1 (gathering ideas)
  + Talk to customer
  + Research of Kali, Python, Electron and other applications and tools
  + Research on how to sync databases, export images and text files
  + How to allow connection between users on a local network
  + How to turn findings into a printable reports//optional
* PHASE 2 prototyping
  + Create a GUI schema
  + Create schema of database
  + Test importing of images
  + Test sync function of database
* PHASE 3 database implementation
  + Implement assigning task
  + Implement findings, including images
* PHASE 4 network implementation
  + Allowing synching of team member’s database to team lead’s database over local network
* PHASE 5 testing
  + Have customer test the system
  + Optimize application according to customer
  + More testing of the system
* OPTIONAL TASK
  + Create a printable report from findings in database

# Schedule of milestones / deliverables

|  |  |
| --- | --- |
|  | Roadmap of development |

Tasks

1

* 1. – Gathering Requirements/Ideas
  2. – Finalizing design of the application
  3. – Create GUI prototype and database schema/implementing database
  4. – Prototyping database syncing
  5. – Basic usable interface
  6. – Adding functionality to frontend
  7. – Implement synchronization
  8. – Finish frontend
  9. – Have customer test the system
  10. – Deliver Application

|  |  |  |
| --- | --- | --- |
| Milestone/Deliverable | Start Date | End Date / Due date |
| Statement of Work Draft | 2-01-2020 | 2-06-2020 |
| Task 1.1 | 2-06-2020 | 2-12-2020 |
| Statement of Work | 2-06-2020 | 2-12-2020 |
| Task 1.2 | 2-12-2020 | 2-14-2020 |
| Requirements Specification Draft | 2-12-2020 | 2-19-2020 |
| Task 2.1 | 2-14-2020 | 2-21-2020 |
| Task 2.2 | 2-21-2020 | 2-28-2020 |
| Requirement Specification Document | 2-19-2020 | 2-29-2020 |
| Task 3.1 | 2-29-2020 | 3-11-2020 |
| Design Specification Draft | 2-29-2020 | 3-17-2020 |
| Task 3.2 | 3-12-2020 | 3-26-2020 |
| Design Specification Document | 3-17-2020 | 3-30-2020 |
| Test Plan Draft | 3-30-2020 | 4-06-2020 |
| Task 4.1 | 3-26-2020 | 4-09-2020 |
| Task 4.2 | 4-09-2020 | 4-23-2020 |
| Test Plan Documentation | 4-06-2020 | 4-13-2020 |
| Task 5.1 | 4-23-2020 | 4-30-2020 |
| Task 5.2 | 4-30-2020 | 5-06-2020 |

# Special requirements

* Requirements
  + Will be running on a laptop, networked through a switch. Kali Linux (Version:? ) May have limited dependencies available to the application, given the restricted environment in which it will run. Must not rely on any external web applications or infrastructure.
* Knowledge / Special Skills
  + Knowledge of databases and use of databases is important. Distributed systems, in particular how to keep data in sync between machines with intermittent connectivity. Some knowledge of web programming will be needed for the front end/ interface of the application. Some network programming will be needed for synchronizing data among machines. (most likely python)
* Restrictions
  + Government restrictions, such as what is allowed to run on the machines.

# Acceptance criteria

The ARL team using the application will be the ones to decide when work is acceptable, since they will be the ones

using the application. If they do not deem it a stable enough tool, then it is not acceptable to be deployed. We, as a

team, will know when it is acceptable, once we have done a demo, and reviewed the application with the team, and

they say it is usable in the field. When we have the next release of the application, we will meet with the ARL team,

to run through the application, and its functionality with them. At this point, any questions about functionality or

implementation can be addressed, for potential reassessment, or for their assurance. Once they deem the application

ready for the field, they can deploy/download the program and use it as necessary.

The most important factors that will decide if the application is ready, is its security, and stability. The notes the application will hold will be very important, and any loss of data will be unacceptable, as well as any leaking of data.

# Reporting

|  |  |
| --- | --- |
|  | Communication strategy with the ARL team |

Due to our customers schedule and travel plans we will communicate with our customer using Slack/e-mail.