Harmonica AMT

Student name: Liam Riche

Student number: B00789989

Course: Computing Systems PT

Date: 2022/2023

Supervisor: Glenn Hawe

Contents

[Abstract 3](#_Toc131092236)

[Acknowledgements 3](#_Toc131092237)

[Listing of Figures 3](#_Toc131092238)

[Listing of Tables 3](#_Toc131092239)

[Listing of Code Listings 3](#_Toc131092240)

[Introduction 3](#_Toc131092241)

[Background 3](#_Toc131092242)

[Project Aims 3](#_Toc131092243)

[Project Objectives 3](#_Toc131092244)

[Outline of Dissertation Structure 3](#_Toc131092245)

[Literature Review 3](#_Toc131092246)

[Existing Solutions 3](#_Toc131092247)

[Current System 3](#_Toc131092248)

[Desired Solution 3](#_Toc131092249)

[Potential Existing Solutions 3](#_Toc131092250)

[Weighing up the options 3](#_Toc131092251)

[Technical Background 3](#_Toc131092252)

[Choosing a Software Development Life-Cycle 3](#_Toc131092253)

[Considerations 3](#_Toc131092254)

[Software Development Life-Cycles 3](#_Toc131092255)

[Choosing a life-cycle model 3](#_Toc131092256)

[Choosing the tools 3](#_Toc131092257)

[Identifying the needs 3](#_Toc131092258)

[Choosing a web framework 3](#_Toc131092259)

[Choosing a database backend 3](#_Toc131092260)

[Development Environments and Other Tools 3](#_Toc131092261)

[Project Planning 3](#_Toc131092262)

[Resource Consideration 3](#_Toc131092263)

[Risk Assessment 3](#_Toc131092264)

[Data Management 4](#_Toc131092265)

[Knowledge and Skills Required 4](#_Toc131092266)

[Involving Stakeholders 4](#_Toc131092267)

[System Requirements 4](#_Toc131092268)

[Requirements Gathering 4](#_Toc131092269)

[Requirements 4](#_Toc131092270)

[Functional Requirements 4](#_Toc131092271)

[Non-Functional Requirements 4](#_Toc131092272)

[Design 4](#_Toc131092273)

[UX Design 4](#_Toc131092274)

[Design Principles 4](#_Toc131092275)

[UI Designs 4](#_Toc131092276)

[UI Implementations 4](#_Toc131092277)

[Data Modelling 4](#_Toc131092278)

[Database Design 4](#_Toc131092279)

[Data Models 4](#_Toc131092280)

[Implementation 4](#_Toc131092281)

[Architecture Overview 4](#_Toc131092282)

[Architecture 4](#_Toc131092283)

[Additional libraries used 4](#_Toc131092284)

[Code Overview 4](#_Toc131092285)

[Obstacles Encountered 4](#_Toc131092286)

[Testing & Evaluation 4](#_Toc131092287)

[Testing 4](#_Toc131092288)

[Unit Testing 4](#_Toc131092289)

[System Testing 4](#_Toc131092290)

[Manual Testing 4](#_Toc131092291)

[Compatibility Testing 4](#_Toc131092292)

[Evaluation 4](#_Toc131092293)

[Evaluation of technologies used 4](#_Toc131092294)

[Evaluation of system 4](#_Toc131092295)

[Results 4](#_Toc131092296)

[Fulfilment of project objectives 4](#_Toc131092297)

[Fulfilment of functional requirements 4](#_Toc131092298)

[Fulfilment of non-functional requirements 5](#_Toc131092299)

[Additional work completed 5](#_Toc131092300)

[Conclusion 5](#_Toc131092301)

[Reflecting on the project 5](#_Toc131092302)

[Project scope 5](#_Toc131092303)

[Products produced 5](#_Toc131092304)

[Managing the process 5](#_Toc131092305)

[Suggestions for future improvements 5](#_Toc131092306)

[References 5](#_Toc131092307)

[Appendices 5](#_Toc131092308)

[Appendix 1 – Initial Gantt Chart 5](#_Toc131092309)

[Appendix 2 – Unit tests listing 5](#_Toc131092310)

# Abstract

# Acknowledgements

# Listing of Figures

# Listing of Tables

# Listing of Code Listings

# Introduction

## Background

The harmonica is a wind instrument played by either inhaling or exhaling a breath on certain parts of the harmonicas front to produce different noises. Harmonicas can have many different tunings, with the most common being in the key of C with 20 possible notes at different pitches. The music notation for the harmonica is also an outlier to other sheet music as instead of the common stave format of 5 horizontal bars commonly seen with piano music, a harmonica music notation is much simpler, with notations often simply being the number of the note and an arrow indication wether the note is a draw or blow.

This project came into being from my interest in Automatic Music Transcription services (AMTs) that take an audio input for a specific instrument and convert it into a sheet music notation. When looking into these I found many AMTs for popular instruments like the guitar and piano, but very few for more niche instruments. I chose to base my project around the harmonica as it is niche enough that there is not an AMT already created for it, as well as the fact that the harmonica has enough unique aspects to it that a tailored solution would offer more to a user compared to a general use AMT

The intended user base for this would be people who play the harmonica and either want to:

1. Find the notes to a song they are listening to
2. Create sheet music for a song they have come up with to share with others

### Project Aims

My main aim for this project is to create an automatic music transcribing program that will take a user input from a user via uploading or recording a music file containing a harmonica piece and gather information about the recording, including:

* tempo (speed at which music is played),
* note length (how long a note is held / a silence lasts)
* pitch of note (which note is being played)
* key (the tuning of the instrument)
* blow direction (is the musician sucking or blowing into the harmonica)

the AMT will then piece these together with a renderer to generate sheet music for the sound file that was uploaded and offer the user a method to view or download the finished product

The software would be created using python with some dataset manipulation to extract important information from the sound file. This python program would be packaged into a website or application to offer a more user friendly method of interacting with the AMT rather than just having it run off the console.

## Project Objectives

At the beginning of this project I set myself a few key objectives to aim to complete during the course of the alotted time:

1. find and understand key requirements that the stakeholders of the project may have
2. generate a set of functional and non functional requirements for the finished program
3. identify and choose the correct software to fit the requirements of the project, as well as choose an effective software development cycle for the projects duration
4. perform a risk assessment of issues that could arise during the development of the project and come up with countermeasures to ensure mitigation of risks
5. research what software would be most effective for creating a final product that would fulfil the shareholders criteria
6. design and use a gantt chart to track and make effective use of my time when working on the project
7. create a product that allows a user to create and download a sheet music version of whatever tune was played into it
8. perform testing with potential users of the product and create a list of feedback and areas of improvement for the future
9. produce a readme for the project that would both teach the user how to use the product as well as how to set up a new instance of the product with ease

## Outline of Dissertation Structure

Chapter 2 – Literature Review & Technical Background

This section will include information detailing the basics of the project and the research into any prior existing products that fulfil similar roles in the market

Chapter 3 – Requirements Gathering & Analysis

This section will cover the gathering, ordering and verification of requirements for the project, as well as the time allotment for the different steps of the project and creating a list of user requirements

Chapter 4 – Project Planning & Preparation ---

This includes details regarding project management tools, version control, risk assessment, chosen software development lifecycle methodology, Gantt chart as well as an overview of the hardware and software utilised.

Chapter 5 – Design

This includes details regarding use case diagrams, screen design phases, colour palettes, accessibility and a discussion regarding the ER Diagram.

Chapter 6 – Implementation & Testing

This includes details regarding the implementation of user stories into features within the application. This highlights the code used to achieve each user story, using unit tests to confirm that user stories are fully met and providing screenshots to provide evidence of this.

Chapter 7 – Evaluation

# Literature Review

## Project foundations

In 2022 Spotify released an open source python module called basic pitch, which allows for a multitude of types of audio files to be passed through it, and using a neural network will output a midi file and an optional csv file with the predicted pitches that were played in said audio file as well as when they were played.

As well as this I had contact with multiple shareholders that played the harmonica and were interested in an application to encode songs they played on a harmonica into a sheet music version of the audio file.

## Existing Solutions

While searching for previously existing harmonica transcribers, I found many systems that offer plain audio to midi and sheet music transcribers (basic pitch is an example of such), however none of these would work correctly for a harmonica, as each note on a harmonica (bar the lowest and highest notes) are a mixture of different notes played at the same time with varying pitches, which would not return an easily recognisable notation format that the harmonica uses.

\*screenshots of audio to sheet notation programs (basic pitch, songify, ect.)\*

## Desired Solution

With the system I aim to create, the returned sheet music will be in the harmonicas sheet music format of a number corresponding to the harmonicas note played and an arrow or other icon / text to indicate the blow or draw direction (this could be done through an up or down arrow, or a negative operator symbol to indicate a draw note and lack of said symbol to indicate a blow note)

\*insert screenshots of harmonica notation and possible draw / blow notation

## Weighing up the options

Upon further researching into existing solutions for harmonica transcribers and general transcribers currently available, the best course of action would be to create an application that uses the existing basic pitch module to process an audio file passed into the applications frontend. The results from this module will then be passed through to the backend for processing that will further fine tune the results from this basic pitch into a json object of harmonica notes. This json object will then be passed back to the applications frontend and rendered in a format recognisable to the user. The rendered notes will then be downloadable for future reference of the user in their own time so that they do not have to use the application to view the generated notes.

# Technical Background

## Choosing a Software Development Life-Cycle

* Agile was chosen for use on the project for its benefits below:
  + Easy and swift change adaption in case of change or oversight in original plans
  + Transparency in current state of project compared to projected progress
  + Allows for continual testing of overall product more often than waterfall model
  + Better stakeholder engagement, updates on current project states more often

### Considerations

### Software Development Life-Cycles

### Choosing a life-cycle model

## Choosing the tools

### Identifying the needs

The proposed system for the harmonica transcriber would be a web app running either locally or hosted online, consisting of a frontend for the user to interact with (uploading files, displaying generated sheet music, downloading sheet music, etc) and a backend that will contain all functions needed to facilitate actions taken on the front end and to process any data or files passed through to the frontend.

### Choosing a web framework

### Choosing a database backend

### Development Environments and Other Tools

# Project Planning

## Resource Consideration

## Risk Assessment

## Data Management

## Knowledge and Skills Required

## Involving Stakeholders

# System Requirements

## Requirements Gathering

## Requirements

### Functional Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Requirement** | **ID** |  |
| R1 | Application must be user friendly and easy to navigate |  |  |
| R2 | application must be able to record or upload sound clip |  |  |
| R3 | Application must return error message if file uploaded is not compatible |  |  |
| R4 | Application must return error message if file uploaded is longer than max length |  |  |
| R5 | Application must be self contained and have ability to function on its own |  |  |
| R6 | Application should return sheet notation of an uploaded clip in pdf format |  |  |
| R7 | Application should have accessible gui menu |  |  |

### Non-Functional Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Requirement** | **ID** |  |
| R1 | Application must be user friendly and easy to navigate |  |  |
| R2 | application must be able to record or upload sound clip |  |  |
| R3 | Application must return error message if file uploaded is not compatible |  |  |
| R4 | Application must return error message if file uploaded is longer than max length |  |  |
| R5 | Application must be self contained and have ability to function on its own |  |  |
| R6 | Application should return sheet notation of an uploaded clip in pdf format |  |  |
| R7 | Application should have accessible gui menu |  |  |

# Design

## UX Design

Graphical user interface, diagram

Description automatically generated

### Design Principles

### UI Designs

### UI Implementations

## Data Modelling

Replace this section with csv file editing or similar

### Database Design

### Data Models

# Implementation

## Architecture Overview

### Architecture

### Additional libraries used

## Code Overview

## Obstacles Encountered

# Testing & Evaluation

## Testing

### Unit Testing

### System Testing

### Manual Testing

### Compatibility Testing

## Evaluation

### Evaluation of technologies used

### Evaluation of system

# Results

## Fulfilment of project objectives

### Fulfilment of functional requirements

### Fulfilment of non-functional requirements

### Additional work completed

# Conclusion

## Reflecting on the project

### Project scope

### Products produced

### Managing the process

## Suggestions for future improvements

# References

# Appendices

## Appendix 1 – Initial Gantt Chart

A picture containing graphical user interface

Description automatically generated

## Appendix 2 – Unit tests listing

## https://blog.pythonanywhere.com/169/