**Indiana University Southeast – CSCI Department**

**CS Capstone [I] – [Preliminary] [Software Detailed Design]**

**Fall 2016**

**Audemes**

**Alan Grant, Jeremy Stephens, Rebecca Luttmer**

The SDD should contain the following information:

Each procedural component should include one section devoted to the detailed description of the component

* Introduction/Purpose of this Component/Entity
* Input for this Component/Entity
* Output for this Component/Entity
* Component/Entity Process to Convert Input to Output
* Design constraints and performance requirements of this Component/Entity
* Process (pseudo-code algorithm)

The product to be produced is a I expect a system specification.  
  
You should be able to pass the document off to another team of very junior programmers and they could build that system.  
  
All programs should be described in detail, list all major methods/routines and the English one sentence description of each method/routine. And all should be very computable (no magic occurs here).  
  
The database (or all data store) should be described at the ER Diagram level.  
  
Products for hardware platform, database and language for all programs should be specified.

Also, this report should be posted to the team website.

Portions of previous assignments can be used within this document in appropriate sections.

Note: Portions of this document are from <https://en.wikipedia.org/wiki/Software_design_description> .

1. Cover page (as described previously)
2. The [*data design*](https://en.wikipedia.org/wiki/Data-driven_design) describes structures that reside within the software. Attributes and relationships between [data objects](https://en.wikipedia.org/wiki/Data_object) dictate the choice of [data structures](https://en.wikipedia.org/wiki/Data_structures).

Here is an overall view of the data flow, from the previous lab:

* 1. Overview



* 1. Subsystem decomposition – Identify the subsystems and the responsibilities of each. You should use component diagrams.



The main data structure will be created using MySQL. It will be a table with the following attributes:

### Audemes Table

Image of Audemes Table and associated fields

* IDNumber - Unique identifier used as key value. Could be the file name if they are unique. Or it could be as simple as an integer value starting from one.
* atomic - Boolean value. True if the list is an atomic Audeme. i.e. an Audeme not composed of other Audemes.
* categories - String value. A delimited list of categories that the Audeme can be considered a part of.
* fileSize - Integer value. The size of the file in bytes. May be useful for some operations.
* fileFormat - String value. A string to simply store the file type. i.e. .wav, .mp3, etc.
* filepath - String value. Stores the needed filepath to access the sound file for the Audeme.

Here are the substructures:

Dictionary DOM (HTML):

The index will have two JavaScript onClick() methods:

|  |  |
| --- | --- |
| Name | onClickSearch() |
| Purpose | Processes the user’s click request |
| Input | String[] keywords, String category  (can be both or either) |
| Output | List of table objects |
| Process | Sends the user input to the request interface |
| Constraints | 0 < String < 50 |
| Pseudocode | onClickSearch(String[] keywords, String category){  ro = new RequestObject(String keywords, String category);  SQLGenerator(ro); |

|  |  |
| --- | --- |
| Name | onClickPlay() |
| Purpose | Processes the user’s click request |
| Input | Int tableIndex |
| Output | An audio file |
| Process | Sends the user input to the play interface |
| Constraints | 0 < String < 50 |
| Pseudocode | onClickSearch(int tableIndex)  Audio file af = Play.get(int tableIndex) |

|  |  |
| --- | --- |
| Name | Request Class |
| Purpose | Creates an organizational object for the user’s request |
| Input | String[] keywords, String category  (can be both or either) |
| Output | Itself |
| Process | Creates an object |
| Constraints | 0 < String < 50 |
| Pseudocode | New Request(String [] keywords, String category)  Fields: keywords, category  Constructor() |

|  |  |
| --- | --- |
| Name | SQLGenerator Class |
| Purpose | Formats a SQL request |
| Input | String[] keywords, String category  (can be both or either) |
| Output | A properly formatted SQL request |
| Process | Turns the strings into a SQL request |
| Constraints | 0 < String < 50 |
| Pseudocode | SQLGenerator(String [] keywords, String category)  Tq = New table query (keyword join all keywords, string category)  Return tq |

|  |  |
| --- | --- |
| Name | TableObject |
| Purpose | Gets information from table and creates organized object. |
| Input | SQL query |
| Output | TableObject  Image of Audemes Table and associated fields |
| Process | Queries SQL and returns the table object |
| Constraints | Must be properly formatted SQL string |
| Pseudocode | TableObject(sql)  Table t = Database.query(sql)  Here we may have to reformat the SQL table?  Return t; |

|  |  |
| --- | --- |
| Name | Play class |
| Purpose | Retrieve the audio file and play it |
| Input | int |
| Output | Audio file |
| Process | Queries the table object for the file location |
| Constraints |  |
| Pseudocode | Play(int)  Tablerow tr = tableObject.get( int tablerow clicked)  File f = tr.getFile(tr.location)  Return f; |

1. The [*architecture design*](https://en.wikipedia.org/wiki/Software_architecture) uses information flowing characteristics, and maps them into the program structure. The transformation mapping method is applied to exhibit distinct boundaries between incoming and outgoing data. The data flow diagrams allocate control input, processing and output along three separate modules.

# Architecture Design

## The [architecture design](https://en.wikipedia.org/wiki/Software_architecture) uses information flowing characteristics, and maps them into the program structure. The transformation mapping method is applied to exhibit distinct boundaries between incoming and outgoing data. The data flow diagrams allocate control input, processing and output along three separate modules.



1. The [*interface design*](https://en.wikipedia.org/wiki/Interface_design) describes internal and external program interfaces, as well as the design of human interface. Internal and external interface designs are based on the information obtained from the analysis model.



1. The [*procedural design*](https://en.wikipedia.org/wiki/Procedural_design) describes structured programming concepts using graphical, tabular and textual notations. These design mediums enable the designer to represent procedural detail that facilitates translation to code. This blueprint for implementation forms the basis for all subsequent software engineering work.



1. Breakdown of individual contributions

Rebecca Luttmer-Data Design, Architecture Design, Procedural Design

Alan Grant-

Jeremy Stephens-Interface Design

1. Key Personnel information

# Personnel

Alan Grant – Software developer

Jeremy Stephens—Software developer

Rebecca Luttmer—Software developer

Heather MacKenzie, Department manager, American Printing House for the Blind—Product owner, Consultant

Larry Skutchan, Department head, American Printing House for the Blind—Product owner, Consultant

Steve Mannheimer, Professor, Indiana University Purdue University Indianapolis—Produce owner, Consultant