*We are developing a web application service designed to allow the average university student more control in their scheduling process.*

*The Reckz*

Project Portfolio

*February 7, 2023*

[Introduction : The Little Webservice That Could 2](#_Toc127962380)

[The “Reckz” Team : The Webservice That Could 5](#_Toc127962381)

[System Requirements : Our System Requirements 7](#_Toc127962382)

[Requirements 7](#_Toc127962383)

[Epics 7](#_Toc127962384)

[Epic #1 7](#_Toc127962385)

[User Stories 7](#_Toc127962386)

[User Story #1 7](#_Toc127962387)

[User Story #2 7](#_Toc127962388)

[User Story #3 7](#_Toc127962389)

[Project Management 8](#_Toc127962390)

[Continuity of Operations Plan (COOP) : Our Operation’s Plan, and You! 8](#_Toc127962391)

[Project Plan 0](#_Toc127962392)

[System Architecture Design and Development Milestone 1->2: **The Milestone** 0](#_Toc127962393)

[System Implementation Milestone 2->3: Our Implementations 1](#_Toc127962394)

[Project Postmortem <Postmortem> 0](#_Toc127962395)

[Project Wins 0](#_Toc127962396)

[Root Cause Analysis 0](#_Toc127962397)

[Lessons Learned 0](#_Toc127962398)

[System Design Milestone 2: Our Webservice Architecture 1](#_Toc127962399)

[System Architecture Milestone 2: Our Webservice Architecture 1](#_Toc127962400)

[Component Design 1](#_Toc127962401)

[Data Flow 2](#_Toc127962402)

[System Components <Milestone 3: System Implementation> 2](#_Toc127962403)

[Component [Component Name 1] 2](#_Toc127962404)

[Component [Component Name 2] 2](#_Toc127962405)

[Component [Component Name n] 2](#_Toc127962406)

[Design Pattern <Milestone 3: System Implementation> 3](#_Toc127962407)

[Design Pattern <Milestone 3: System Implementation> 3](#_Toc127962408)

[System Implementation <Milestone 3: System Implementation> 4](#_Toc127962409)

*All diagrams developed using Enterprise Architect are thanks to SPARX Systems for allowing LSU students and faculty to use of Enterprise Architect for free in regards to academic purposes”.*

# Introduction : The Little Webservice That Could

We have observed compounded issues regarding the scheduling process at LSU, and it’s disregard for visual quality over function, along with actual missing features that our team believes *must* be there. The Schedule Request lacks interactive features for checking classes, quick response times, and smaller details such as specific class-availabilities and easily accessible professor ratings via RateMyProfessor.com and other website sources.

We aim to utilize a Web Server to host our prototype, with a backend programmed in Visual Studio and/or VS Code in the C# Programming Language. Our implementation will be done first through a simple visual representation of the basic necessities of interactive schedule management, where the backend code will only support the most necessary of data. The server itself will be an Amazon EC2 Windows Web Server Instance.

|  |  |
| --- | --- |
| Visual Studio logo and symbol, meaning, history, PNG | With over 1.3 million professors, 7,000 schools & 15 million ratings, Rate  My Professors is the best profes… | Rate my professor, Feedback for  students, College fun |
|  | Logo Html Html5 - Free image on Pixabay |
|  | Step by Step Creation of an EC2 Instance in AWS and Access it via… –  Towards AI |
| JavaScript logo and symbol, meaning, history, PNG | Logo Css Css3 - Free image on Pixabay |

Core Features:

* Display of Professor Information
* Clear & Concise Waitlist
* Display of Class’ Student Count & Class Statuses
* Accessible via Website
* Visual Representations of Class Hours
  + Arrow to reveal dropdown of Schedule
* Accessible by Windows, MacOS, Linux & IOS
* Interactable & Adjustable “Current Schedule” In User’s Immediate Access

Viable Features:

* Display Professor Information
* Visual Representation of Class Data
* Dropdown/Combobox Menu
* Accessible Via Website in a User-Friendly Fashion
* A Single Login Required

Stretch Features

* Login To LSU Account *once* to Access Schedule Data in LSU’s Database
  + Will need to contact LSU’s IT
* The Display of Prerequisite of a given class
* Porting the Application for Mobile Use
  + Simplified Design required
* Addition quality of life features such as financial documentation
* Easy-access todo-listing for classes
* Exam-Times-date-manager
  + Works based off of given classes, not just any

# The “Reckz” Team : The Webservice That Could

The team’s structure is semi-democratic in form, with most of the overall project direction being made by the Project Lead, with each of the team contributing and influencing the direction according to the overall general consensus of the group. All disagreeances made within the group to the point of a standstill can be overwritten by the project lead, whom is responsible for the coordination and subsequent encouragement of consistent communication between all members of the team whom are each responsible for their individual initiatives to contribute to the project.

The project lead may pick any given member to take their place if the group deems it that they are unsuited to lead a current milestone, however the group could also simply waver the project lead’s choice and instead elect a different member at any respectable time. If all members disagree, then the project’s mentor will need to pick.

**Elias Shalohm** [*Project Lead*]

Is the main driving force for keeping the team in check and establishing a consistent scheduling basis; coordinating respective efforts to meet the end goal. They are responsible for documenting the group’s ongoing progress as a representative of the team. This member also acts as one of the two primary programmers acting as support for the rest of the team.

* Covers the “Look and Feel” of the application; aiming for user interactivity and user-friendliness.
* Visual Representations of Class Hours
* Arrow to reveal dropdown of Schedule

**Seif Sultan** [*Auxiliary Developer*]

Will act as a substitute for Project Lead given the instance that the Project Lead is either missing attendance, or has failed in their role.

* Covers the Webserver’s JavaScript and data management.
  + Display of Professor Information
  + Accessible via Website
* The Display of Prerequisite of a given class

**Kyle Jeffeson** [*Auxiliary Developer*]

As a Documenter; will be documenting changes in the project’s code, reviewing the Github commits to ensure quality, updating the project spreadsheet and making important adjustments to the Project Milestone Proposals.

* Covers the backend of the Web-Application’s code and the interactions between Appearance & Data.
* Display of Class’ Student Count & Class Statuses

**Cameron Olivier** [*System Architect*]

Organizes the architectures of the project, acting as the primary driver for the team’s development direction.

* Clear & Concise Waitlist
* Login To LSU Account once to Access Schedule Data in LSU’s Database
  + Will need to contact LSU’s IT

**Rayhoar Ebrahim** [*Auxiliary Developer*]

Handles the Intractability and adjustability of the User’s “Current Schedule” which should be Immediately Accessible.

**Donald Saulny** [*Forefront* *Designer*]

Is the priority-designer for the project; whom creates the necessary design plans for the rest of the team to collaborate directly-with. This member also acts as one of the two primary programmers acting as support for the rest of the team.

* Accessible by Windows, MacOS, Linux & IOS

**Project Github:** <https://github.com/EliasShalohm/RecksProject>

# System Requirements : Our System Requirements

## Requirements

Our system requirements will consist of the user’s access to the internet with a Bandwidth supported by machines preferably made after the year 2000. The OS basis required to run the program should be Windows 2000, with at least 10GB of Harddrive space, and 3 GB of Ram. Users should have access to the respective peripherals needed to access the website.

## Epics

### Epic #1

As a University Student, I want to use a website in such a way, so I can manage my schedule.

## User Stories

### User Story #1

As a Computer Science Student attending Louisiana State University, I want to have direct access to my schedule’s individual classes and their hours, so I can pick which ones are necessary to continue my up-and-coming semester.

### User Story #2

As a Professor sponsoring my PhD Student, I want to assist in organizing their high-level classes by class-level and enroll count, so I can be certain that they are not spending as many hours scheduling.

### User Story #3

As an incoming Freshmen at Southern-Southeastern University, I wish to quickly create a visual representation of my schedule for print-outs, so that I can share them with my friends to organize our own club.

# Project Management

## Continuity of Operations Plan (COOP) : Our Operation’s Plan, and You!

The team is to consistently organize and maintain their own schedules on individual basis, where all changes to said-schedules are sent to project lead for syncing between the others’ schedules. All hours are to be coordinated in such a manner that all team members can not only see the merged schedules, but make any given changes deemed necessary.

The team also will be meeting weekly on a group-chosen day as opposed to a constant day-and-hour due to the volatility of personal individual schedules. Anyone missing any of the meetings is immediately filled in on all the discussions covered in the meeting as a major proponent for the group as a whole is fast, if not instantaneous communication & response.

## Project Plan

### System Architecture Design and Development Milestone 1->2: **The Milestone**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Activity** | **Perquisites #** | **Estimated Hours** | **Actual Hours** | **Estimated Start Date** | **Estimated Finish Date** | **Actual Start Date** | **Actual Finish Date** |
| 1 | Backend Infrastructure Development |  |  |  |  |  |  |  |
| 1.1 | Get Temporary Access to LSU Database |  | 15 Hours (Thanks, IT) | 2 Hours | 02/06/2023 | 02/09/2023 | 02/06/2023 | 02/06/2023 (No Thanks To IT) |
| 1.3 | Research Queries for RateMyProfessor |  | 8 Hours | 21 Hours | 02/08/2023 | 02/12/2023 | 2/14/2023 | 02/28/2023 |
| 2 | Webserver Initialization |  |  |  |  |  |  |  |
| 2.1 | Rent / Obtain EC2 (or otherwise) Server |  | 3 Hours | 2 Hours | 02/06/2023 | 02/06/2023 | 02/06/2023 | 02/06/2023 |
| 2.3 | Begin Template on Github | 2.1, 2.2 | 15 Hours | 5 Hours | 02/13/2023 | 02/17/2023 | 02/08/2023 | 02/10/2023 |
| 3 | Visual Representation |  |  |  |  |  |  |  |
| 3.2 | Test Query Class Data | 3.1 | 6 Hours | 15 Hours | 02/16/2023 | 02/19/2023 | 02/08/2023 | 02/30/2023 |
| 3.3 | Test Query Professor Data | 3.1, 3.2 | 7 Hours | 10 Hours | 02/16/2023 | 02/19/2023 | 02/08/2023 | 02/30/2023 |
| 3 | Visual Representation |  | 6 hours | 10 Hours | 02/15/2023 | 02/20/2023 | 02/13/2023 | 03/20/2023 |
| 4 | System Organization |  |  |  |  |  |  |  |
| 4.1 | System Architecture Overall Diagram |  | 10 hours | 3 Hours | 02/14/2023 | 02/20/2023 | 02/14/2023 | 02/28/2023 |
| 4.2 | System Architecture Specific Components |  | 4 hours | 3 Hours | 02/15/2023 | 02/21/2023 | 02/15/2023 | 02/28/2023 |
| 4.3 | Webservice Design |  | 8 hours | 8 Hours | 02/17/2023 | 02/22/2023 | 02/16/2023 | 02/28/2023 |

### System Implementation Milestone 2->3: Our Implementations

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Activity** | **Pre #** | **Estimated Hours** | **Actual Hours** | **Estimated Start Date** | **Estimated Finish Date** | **Actual Start Date** | **Actual Finish Date** |
| 1 | Backend Infrastructure Development |  | 7 hours |  | 02/23/2023 | 02/28/2023 |  |  |
| 1.1 | Collect Class Data (Count, Section, Name, Etc.) | 1.1 | 5 hours | 10 Hours | 02/24/2023 | 02/29/2023 | 02/14/2023 | 03/01/2023 |
| 1.2 | Display ratemyprofessor data |  | 10 hours | 49 Hours | 02/25/2023 | 03/03/2023 | 02/14/2023 | 04/10/2023 |
| 1.3 | Make Webserver Public-Access |  | 4 hours | 10 Hours | 02/23/2023 | 02/27/2023 | 02/23/2023 | 03/15/2023 |
| 1.4 | Establish Data->Server via Github Commit | 2.1, 2.2 | 5 hours | 2 Hours | 02/26/2023 | 03/03/2023 | 02/23/2023 | 03/09/2023 |
| 1.5 | Implement Navigable Classes (Seif’s Experience) | 1.1 | 12 hours | 20 Hours | 02/28/2023 | 3/5/2023 | 02/23/2023 | 3/1/2023 |
| 2 | Frontend Development |  |  |  |  |  |  |  |
| 2.1 | Basic Navigable Website via In-Build Tabs |  | 12 hours | 3 Hours | 02/30/2023 | 03/07/2023 | 02/30/2023 | 03/01/2023 |
| 2.2 | Queryable Class Data | 2.1 | 8 hours | 10 Hours | 02/27/2023 | 03/01/2023 | 02/30/2023 | 03/04/2023 |
| 2.3 | Queryable Professor Data | 2.1, 2.2 | 8 hours | 49 Hours | 02/27/2023 | 03/01/2023 | 04/02/2023 | 04/10/2023 |
| 2.4 | Display Class Enrollment Data | 2.1, 2.2, 2.3 | 6 hours | 4 Hours | 03/01/2023 | 03/10/2023 | 02/30/2023 | 03/02/2023 |

## Project Postmortem <Postmortem>

### Project Wins

[Provide a bulleted list of at least 3 positive aspects of the project.]

### Root Cause Analysis

[Provide a bulleted list of at least 3 negative aspects of the project. For each negative, provide the answer to the three successive “Why” questions. ]

### Lessons Learned

[For each negative aspect identified in the Root Cause Analysis, provide a mitigation strategy (i.e., what process should be introduced) to ensure that the problem is not repeated in subsequent projects.]

# System Design Milestone 2: Our Webservice Architecture

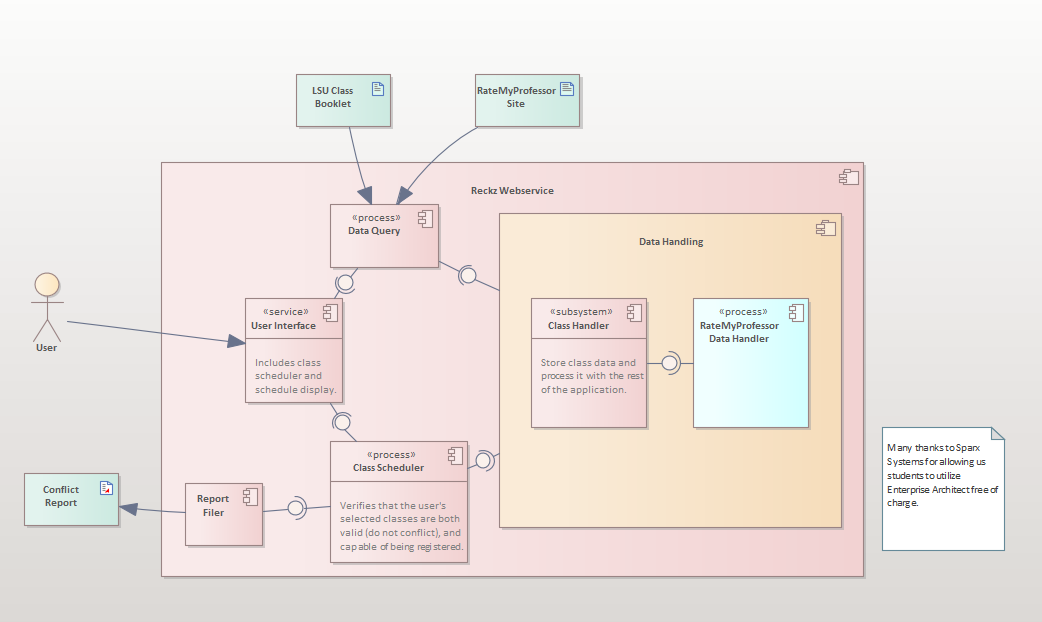
Our architecture is designed in such a way to allow for user input to query class data from LSU’s booklet of Classes. This data is then sent to be handled in our Data Handler, which pulls from RateMyProfessor before the Class data is ultimately “compiled” and organized back into the User interface from the Class Schedule.

## System Architecture Milestone 2: Our Webservice Architecture

The architecture of our system shows how the user’s input into the interface interacts with our Blazor Webservice which extracts data from the LSU Classes Booklet which in turn is processed into instantiated class data. This data is processed back so that the user may see the classes in a scheduled-format.

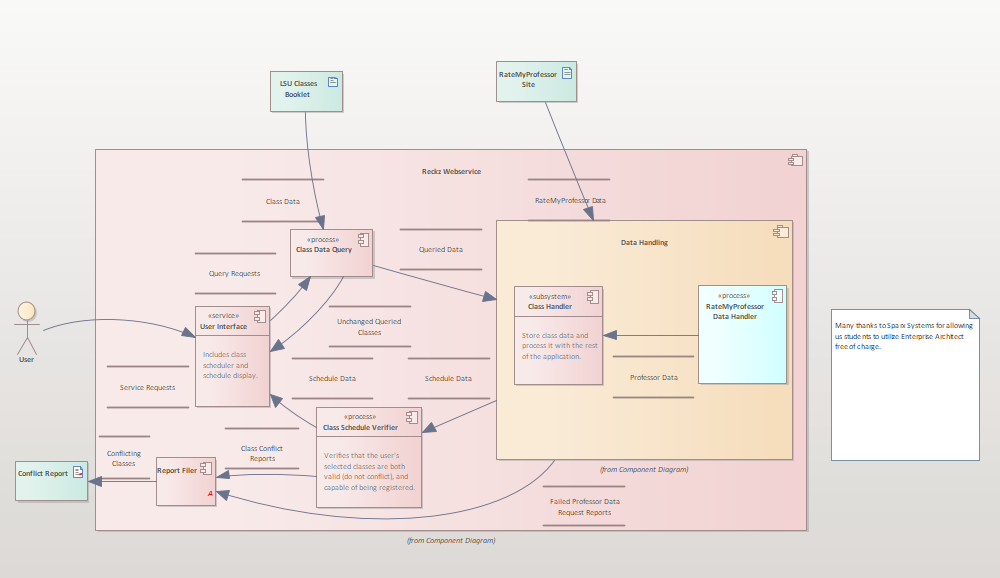
### Component Design

Elias Shalohm



### Data Flow

Elias Shalohm, Cameron Olivier & Seif Sultan

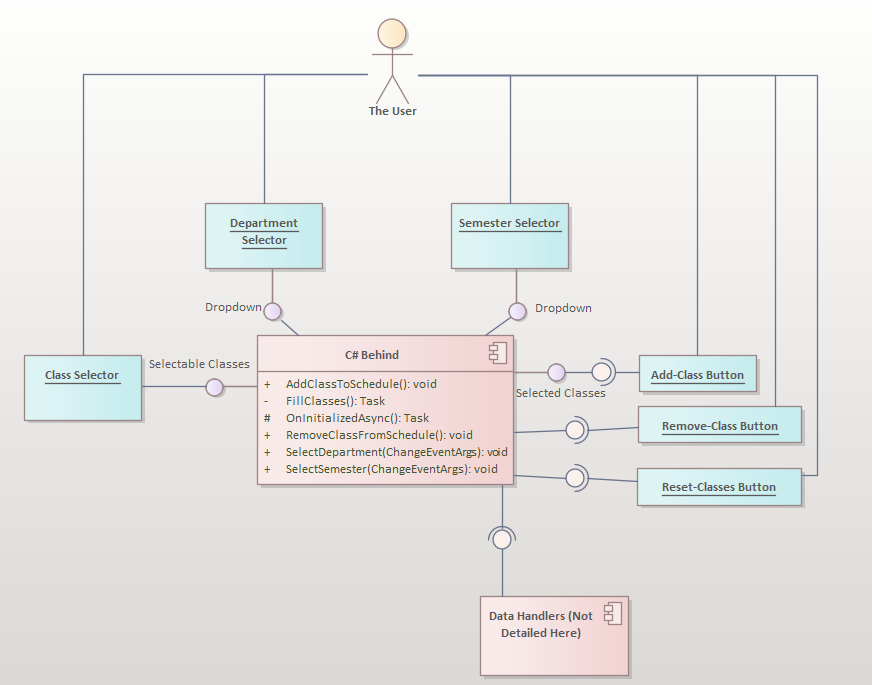


## System Components Milestone 3: Our System Implementation

### Component [User Interface]

The User Interface allows the user to select a Semester and Department indirectly from LSU. With a check made in-component to ensure the user has selected both, and only when the user has selected both, is the class-options dropdown list filled.

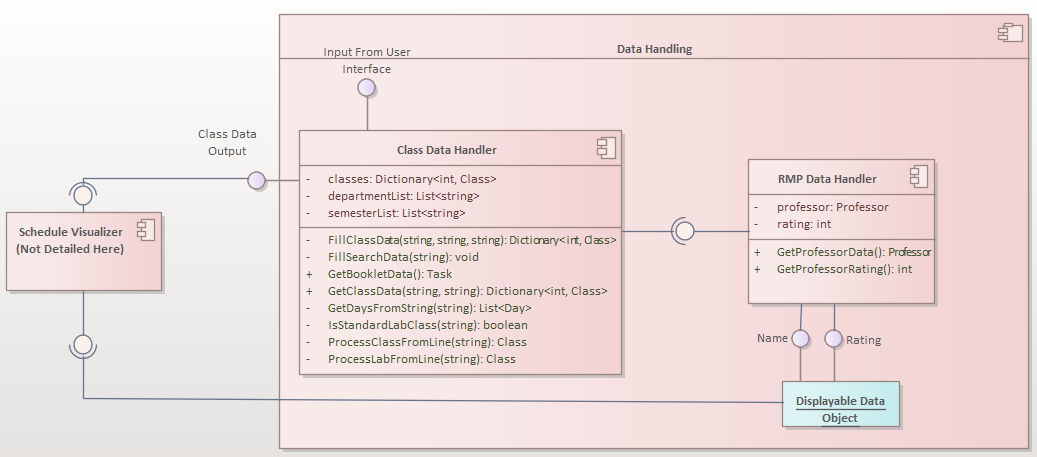
Diagram by Elias Shalohm:



### Component [Data Handling]

The Data Handler is divided into the Class Data Handler and the RMP Data Handler, the latter of which is too small to be worth separating into its own diagram, but still has a role in the program. The Data Handling component is where all of the calculating methods for finalizing the data is done. The data is then output as a set of program-readable strings for the Schedule Visualizer to display.

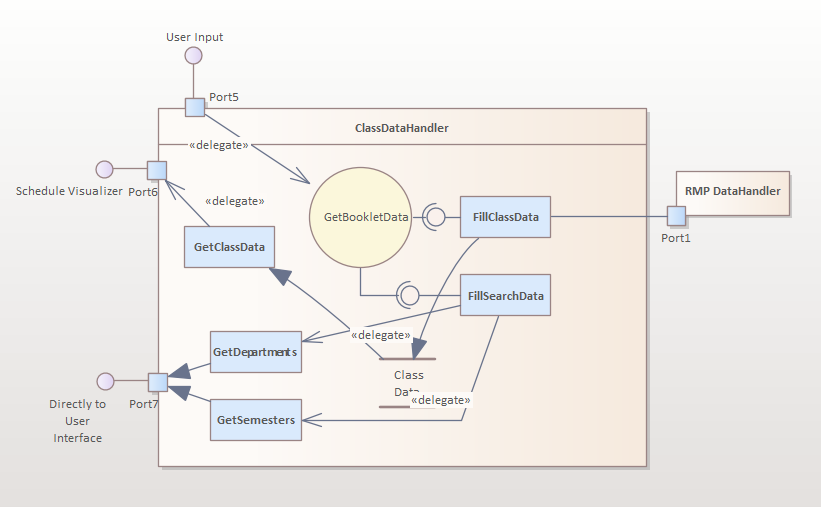
Diagram by Elias Shalohm:



### Sub-Component [Class Data Handler]

Not to be confused with the definition “**class**” in C#, the Class Handler is a component that manipulates the raw booklet data provided it has gained user input, then formats the data into a list of a custom type “**Class**” before outputting the Class Data into the Visualizer, and the updated Booklet data back to the user.

Diagram by Elias Shalohm:



## Design Pattern [Singleton] : Donald Saulny

[*Class diagram of design pattern incorporated into the project. Pattern must be specific to the project and not a general design pattern class diagram. The project must include at least design patterns covered in class. Include the name of the team member that created the diagram in EA.*]

D\_ ClassDataHandler, RMPDataHandler

## Design Pattern [] : Donald Saulny

[*Class diagram of design pattern incorporated into the project. Pattern must be specific to the project and not a general design pattern class diagram. Include the name of the team member that created the diagram in EA. A second design pattern may be included for bonus points.*]

# System Implementation <Milestone 3: System Implementation>

[*In the table below, include a row for each component in your System Architecture diagram. In the second column, list the programming language(s) used to implement the component and the what % of that programming language is used in the implementation. In the third column, list the team member(s) that implement the component and what % of that implementation was completed by that team member. IMPORTANT NOTE: All architectural components must be implemented by an object-oriented programming language: Java, C++, or C#.*]

|  |  |  |
| --- | --- | --- |
| **Architectural Component** | **Programming Language(s) %** | **Team Member(s) %** |
| *[Data Manager]* | *[C++ (45%)*  *Java (55%)]* | *[Mickey Mouse (15%)*  *Donald Duck (20%)*  *Daisy Duck (40%*  *Pluto (25%)]* |