# Detecting and Predicting Clusters of Evolving Binary Stars

DESIGN DOCUMENT

sdmay21-30
Goce Trajcevski (Client & Advisor)
Adam Corpstein (Report Manager),
Becker Mathie (Chief Engineer),
Ethan Vander Wiel (Test Engineer),
Joel Holm (Facilitator),
Philip Payne (Quality Assurance),
Willis Knox (Scribe)
sdmay21-30@iastate.edu
Team Website
http://sdmay21-30.sd.ece.iastate.edu/
Revised: 10/4/2020 -- 1.0

# Executive Summary

## Development Standards & Practices Used

- Agile development
- Project Roles
    - Scribe, Facilitator, Chief Engineer, Test Engineer, Report Manager
- Utilization of Azure Devops for project status and task management
- Confluence for lasting information and project record keeping
- Slack for temporary information and general communication
- Version control using Git with Branch-Review-Merge workflow

## Summary of Requirements

- The product shall be a web based system accessed through an internet connection
- The product shall allow the user to enter initial conditions for the binary system and constraints on that system such as:
    - parameters of interest for clustering
    - distance functions across different attributes, with weighted combinations
    - An interval of interest, expressed as an explicit interval, or an event based interval
- When given initial conditions and constraints for a binary system, the product shall predict whether the system will merge a particular cluster of other known binaries during its evolution
- The product will maintain its own database and eventually allow for the user to input their own database and datasets.

## Applicable Courses from Iowa State University Curriculum

- S E/Com S 309: Software Development Practices
- S E 329: Software Project Management
- S E 339: Software Architecture and Design
- S E 409: Software Requirements Engineering
- Com S 227/228: Introduction to OOP and Data Structures
- Com S 319: Construction of User Interfaces
- Com S 363: Introduction to Database Management Systems
- ENGL 314/309: Technical Communication/Proposal and Report Writing

## New Skills/Knowledge acquired that was not taught in courses

- React.js
- PostgreSQL
- Python/Django
- Clustering Algorithms (K-Means, DBScan)
    - sklearn Python module
- Figma
- Material UI

## Table of Contents

## List of figures/tables/symbols/definitions (This should be the similar to the project plan)

# 1 Introduction

## 1.1 Acknowledgement

The team would like to thank the Electrical and Computer Engineering department at Iowa State University for the resources and education we have received throughout our time spent at the university. We would also like to thank Professor Goce Trajcevski for having weekly meetings with the team to provide feedback and advice.

## 1.2 Problem and Project Statement

### Problem Statement

As astrophysicists continue to collect data on binary star systems in our galaxy, new possibilities for research emerge. Large databases exist full of binary star data. This includes attributes such as red shift, mass ratio, hydrogen ratio, etc. Astrophysicists would like the ability to organize this data into clusters so that any two systems with similar attributes are in the same cluster and dissimilar systems are in different clusters. Astrophysicists have also computed their own data for future states of binary systems. This would add the dimension of time to the data (since the lifetime of a binary star could be millions of years, simulating future states is necessary). By organizing clusters by time, further analysis can be taken as binary stars mature.

### Solution Approach

The driving force for this project is to organize data on binary star systems to aid astrophysicists in their research. To solve the problem of organizing astronomical data, we will apply pre-existing clustering algorithms to the datasets. The results of different clustering algorithms will be available on a web server. From the webpage astrophysicists will select their desired attributes and cluster weight functions to generate binary star clusters. We hope to create and maintain a web server for astrophysicists to use. Also we would like to receive feedback from them on what we can do to improve the process of generating clusters.

## 1.3 Operational Environment

The application will be entirely web-based, so its operational environment is the internet itself. The application will need a reliable connection to the internet for long periods of time to allow for the simulations and calculations to complete.

## 1.4 Requirements

### Functional Requirements

- A web based application with a user interface that accepts different types of data input by users for tracking the stellar evolution of binary stars. Types of data will include but not be limited to:
    - Luminosity
    - Mass
    - Relative distance between stars
    - Helium concentration
- Users will be able to configure which parameters are of interest for clustering
- Users will be able to apply specific weights to their selected parameters that determine the level of importance of each parameter when clustering
- Users will have the ability to specify specific intervals of interest for clusters
    - This interval could either be an explicit time range or a more general event based range
- Given the user specified information, a remote server will access binary stars from a database and then compute and predict how different binary stars will cluster during their evolution

### Environmental Requirements

- The application will be required to work only in areas with an internet connection. This connection will need to be constant and stable to allow users to access the application at any time
- The remote server will need to have the capability to connect to multiple databases at once

### Economic Requirements

- Laptops or personal computers will be needed to interact with the user interface of the application

## 1.5 Intended Users and Uses

The end users for this project will be astrophysicists who have a need to search for similarities between binary star systems. These astrophysicists are used to working with astronomical data provided by Sloan Digital Sky Survey or the Gaia Archive. They may be familiar with the query language SQL as it’s used in these websites.

## 1.6 Assumptions and Limitations

We have come up with the following assumptions and limitations to refine the scope of the problem statement.

### Assumptions

#### *The User can Connect to the Internet*

We are assuming that our intended users will have access to the internet. Without an internet connection, a user will not have the capability to access the application, and therefore they will be unable to use it.

#### *The User has a basic understanding of clustering techniques and weighting functions.*

We are assuming that users have enough knowledge about the provided clustering techniques to understand how it affects the output data. Related to this, the user will need to provide their own weights in regards to each attribute. This also affects the output data, so users will need a solid grasp of how to write their own weighting functions.

### Limitations

#### *The Application should not Overuse the Client's Hardware*

All algorithmic work such as data normalization and star clustering will need to be completed on the server and not on the user's device. This is to ensure the application stays responsive and easy to use.

#### *Types of Clustering Data is Limited to what is in the Pre-Existing Databases*

The different types of data the users will be able to select for clustering will be limited to what information is stored in the databases the application will be accessing. Since we are connecting to well established, pre-existing databases, the users must restrict their clustering queries to types of data that these databases currently contain.

## 1.7 Expected End Product and Deliverables

#### *Website that provides a UI to enter custom cluster queries*

The main feature of this product is the website where astrophysicists will input queries to generate clusters of binary star data. The custom cluster query interface will include the following elements: selection of attributes, input of weight functions, selection of clustering techniques, and selection of the database. The attribute selections allows the user to specify which data types will be regarded in the clustering. Weight functions allow users to specify how heavily each attribute will affect the clustering. Selecting a clustering technique will change how groups of clusters are formed. Lastly, a database must be selected, from which binary star data is pulled. To be delivered April 15th, 2021.

### *Graphical representation of the clustering data after query*

After a clustering query is processed each binary star will be assigned to a cluster. To gain insight into how clusters were created, a graphical representation of the relationship between attributes can be generated. Users will be able to select which two attributes to compare; this will generate a 2D graph showing the clusters based on the two attributes. To be delivered April 15th, 2021.

### *Implementation of clustering algorithms: K-means clustering and DBScan clustering*

There are two clustering algorithms that will be implemented: K-means clustering and DBScan clustering. These two algorithms have their advantages. K-means clustering is an iterative algorithm that refines the cluster formations over time. The main advantage of this approach is its efficient execution time with large datasets. DBScan is a density based clustering technique. Unlike K-means, DBScan does not require a predefined number of clusters. It also is able to create flexible cluster shapes and neglects the impact of outliers. To be delivered April 15th, 2021.

### *User guide page*

To aid users who are unfamiliar with binary star data, clustering techniques, or the process of creating a cluster query, a user guide page will be created. This page will include directions on how to interface with the UI to generate desired clustering output. To be delivered April 15th, 2021.

# 2 Project Plan

## 2.1 Task Decomposition

Iteration 1 (9/10 - 10/10):

- Familiarize with clustering algorithms
  - Kmeans & Dbscan
- Consider possible software tools and platforms

Iteration 2 (10/10 - 10/25)::

- Finalize attributes and distance functions
  - algorithm and distance function research

Iteration 3 (10/25 - 11/10):

- Finalize selection of development platforms and architecture design
- Preliminary UI design

Iteration 4 (11/10 - 11/20):

- Finalize selection of algorithm solutions
- Devise use cases and test cases
  - Use case diagram
  - Test planning

- Finalize UI functionality

Final deliverable (11/20):

- Prepare design presentation
- Finalize and submit design document

## 2.2 Risks And Risk Management/Mitigation

Below are potential risks for this project along with their respective probabilities and mitigation strategies:

- Risk: The team is unable to finish our deliverables in accordance with our timeline.
  - Probability of occurrence: 0.10
- Risk: The client requests that an additional clustering algorithm be added to the project.
  - Probability of occurrence: 0.25
- Risk: The client wants to connect to a database that was not previously provided and has unknown properties.
  - Probability of occurrence: 0.60
  - Mitigation plan:

    To handle the occurrence of this risk our team will design our application to accept various databases. We will additionally implement methods to check the compatibility of the database with our chosen clustering algorithm.
- Risk: The client requests that the cluster data be displayed with a different method that may not be compatible with how our cluster data is returned.
  - Probability of occurrence: 0.50
  - Mitigation plan:

    Similar to the mitigation plan for the previous risk, to limit the possibility of this risk our team will design the application such that it can handle displaying the data in various ways. We will also use design methods that allow for easy changes to the code that handles displaying cluster data without requiring changes to the other parts of the application.
- Risk: A team member contracts COVID-19 and is either unable to work or is only able to do limited work for a period of time.
  - Probability of occurrence: 0.50

- Mitigation plan:

  To limit the possibility of this risk the team is following recommendations for staying safe during the current pandemic. We currently hold all meetings virtually to eliminate the risk of transmission between team members. If a team member contracts COVID-19, the remaining team members will work together to carry out the sick member's responsibilities while they recover.

## 2.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

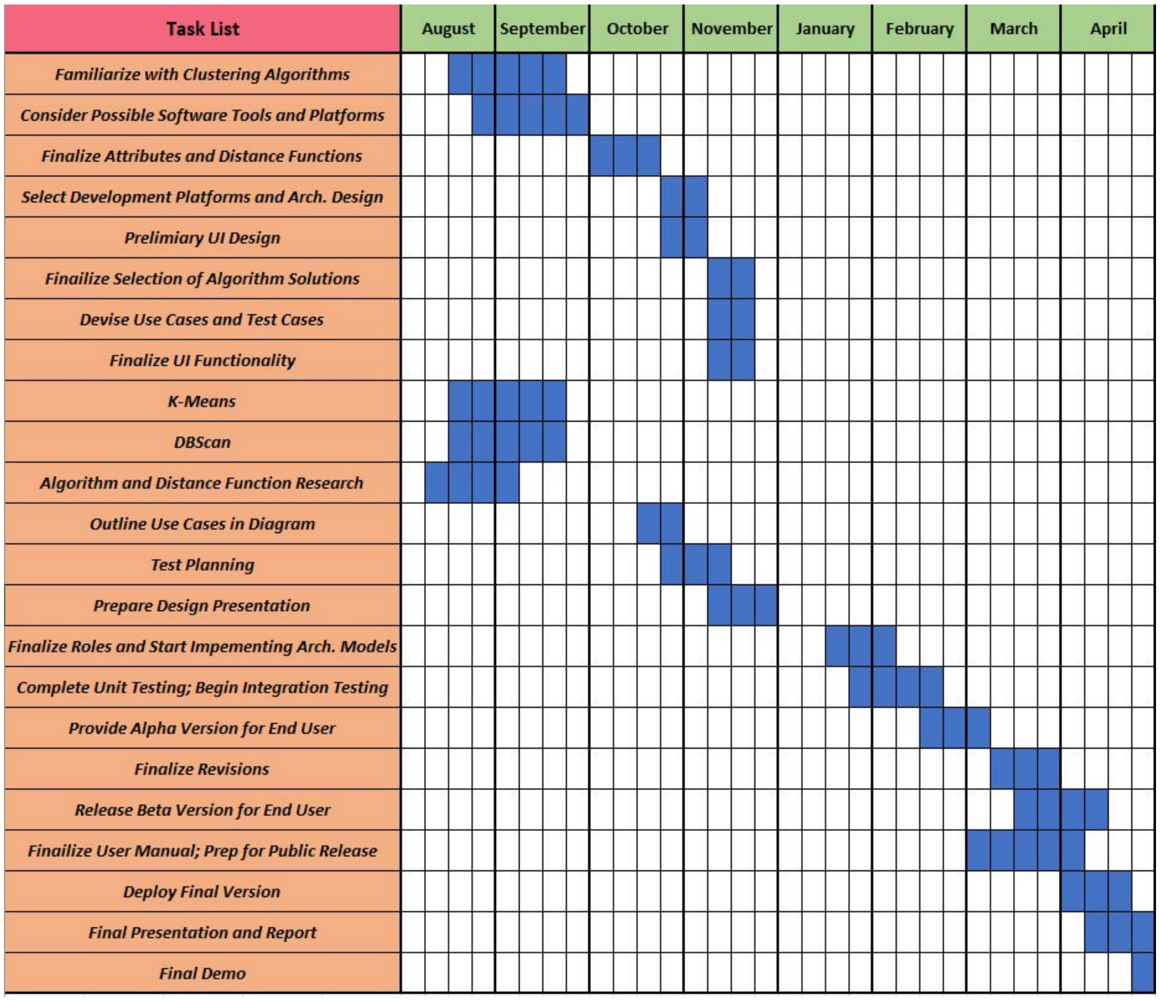
Our project follows two major milestones, the submission of our design document and the presentation of our implementation of the design. The design associated tasks have an emphasis on project requirements and use cases.

- Team understanding of binary star system clustering
- Defining functional requirements
- Outlining the best software tools/platforms for our project
- Determining project usage of algorithms and distance functions
- Defining use cases
- Designing a UI to cover use cases

The criteria above that is connected to the design phase of our project will be evaluated by how well our design covers the requirements of the original project.

The second phase of our project where we will implement our design will have tasks connected directly to the software development. Functional tests will be used to evaluate how well our implementation fulfills the design. Working towards this second milestone will also involve making improvements to non-functional requirements such as reliability, performance, and scalability.

## 2.4 Project Timeline/Schedule



| Task List | August | | | | September | | | | October | | | | November | | | | January | | | | February | | | | March | | | | April | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *Familiarize with Clustering Algorithms* | | | 🟦 | 🟦 | 🟦 | 🟦 | 🟦 | | | | | | | | | | | | | | | | | | | | | | | | | |
| *Consider Possible Software Tools and Platforms* | | | | 🟦 | 🟦 | 🟦 | 🟦 | 🟦 | | | | | | | | | | | | | | | | | | | | | | | | |
| *Finalize Attributes and Distance Functions* | | | | | | | | | 🟦 | 🟦 | 🟦 | | | | | | | | | | | | | | | | | | | | | |
| *Select Development Platforms and Arch. Design* | | | | | | | | | | | | 🟦 | 🟦 | | | | | | | | | | | | | | | | | | | |
| *Prelimiary UI Design* | | | | | | | | | | | | 🟦 | 🟦 | | | | | | | | | | | | | | | | | | | |
| *Finailize Selection of Algorithm Solutions* | | | | | | | | | | | | | | 🟦 | 🟦 | | | | | | | | | | | | | | | | | |
| *Devise Use Cases and Test Cases* | | | | | | | | | | | | | | 🟦 | 🟦 | | | | | | | | | | | | | | | | | |
| *Finalize UI Functionality* | | | | | | | | | | | | | | 🟦 | 🟦 | | | | | | | | | | | | | | | | | |
| *K-Means* | | | 🟦 | 🟦 | 🟦 | 🟦 | 🟦 | | | | | | | | | | | | | | | | | | | | | | | | | |
| *DBScan* | | | 🟦 | 🟦 | 🟦 | 🟦 | 🟦 | | | | | | | | | | | | | | | | | | | | | | | | | |
| *Algorithm and Distance Function Research* | | 🟦 | 🟦 | 🟦 | 🟦 | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| *Outline Use Cases in Diagram* | | | | | | | | | | | 🟦 | 🟦 | | | | | | | | | | | | | | | | | | | | |
| *Test Planning* | | | | | | | | | | | | 🟦 | 🟦 | 🟦 | | | | | | | | | | | | | | | | | | |
| *Prepare Design Presentation* | | | | | | | | | | | | | | 🟦 | 🟦 | 🟦 | | | | | | | | | | | | | | | | |
| *Finalize Roles and Start Impementing Arch. Models* | | | | | | | | | | | | | | | | | | | 🟦 | 🟦 | 🟦 | | | | | | | | | | | |
| *Complete Unit Testing; Begin Integration Testing* | | | | | | | | | | | | | | | | | | | | 🟦 | 🟦 | 🟦 | 🟦 | | | | | | | | | |
| *Provide Alpha Version for End User* | | | | | | | | | | | | | | | | | | | | | | | 🟦 | 🟦 | 🟦 | | | | | | | |
| *Finalize Revisions* | | | | | | | | | | | | | | | | | | | | | | | | | | 🟦 | 🟦 | 🟦 | | | | |
| *Release Beta Version for End User* | | | | | | | | | | | | | | | | | | | | | | | | | | | 🟦 | 🟦 | 🟦 | 🟦 | | |
| *Finailize User Manual; Prep for Public Release* | | | | | | | | | | | | | | | | | | | | | | | | | 🟦 | 🟦 | 🟦 | 🟦 | 🟦 | | | |
| *Deploy Final Version* | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 🟦 | 🟦 | 🟦 | |
| *Final Presentation and Report* | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 🟦 | 🟦 | 🟦 |
| *Final Demo* | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 🟦 |

*Figure 1: Project Gantt Chart*

The chart above shows the time frames associated with our tasks and deliverables for both semesters one and two. The gap indicates the winter break between the semesters. The first semester has tasks working towards our design to be implemented in the tasks layed out for the second semester.

The team member primarily responsible for each task will follow our team roles of Scribe, Facilitator, Chief Engineer, Test Engineer, and Report Manager. Our team has decided that we will be flexible in contributing outside of our roles, and will likely switch roles along the way.

## 2.5 Project Tracking Procedures

We plan to use standard Agile and Scrum styled boards to track our progress and current goals. We will be using Azure Devops for our project and using their built in task and story tracking. To follow Agile standards, our board will have a backlog, to-do on the current sprint, items we are currently working on, in review items, and finished columns. We will progress tasks and stories through these columns to keep track of how much work we have done and what our current priorities are.

As tasks are added from our advisor and our own grooming we will define the tasks on our backlog. Each sprint, tasks from the backlog will be moved to the to-do column and become our sprint priorities.

To communicate current progress on tasks we will discuss in Slack and weekly meetings.

## 2.6 Personnel Effort Requirements

| Task | Iteration Length | Person Hour Estimate | Team Members Needed | Total |
|---|---|---|---|---|
| Familiarize with clustering algorithms | 30 Days | 5 each | 6 | 30 Hours |
| Consider possible software tools and platforms | 30 Days | 5 each | 6 | 30 Hours |
| Finalize attributes and distance functions | 15 Days | 10 each | 6 | 60 Hours |
| Initial UI Design | 16 Days | 5 each | 3 | 15 Hours |
| Finalize selection of development platforms and architecture design | 16 Days | 10 each | 4 | 40 Hours |
| Finalize selection of algorithm solutions | 10 Days | 5 each | 3 | 15 Hours |
| Devise use cases and test cases | 10 Days | 5 each | 3 | 15 Hours |
| Finalize UI functionality | 10 Days | 5 each | 3 | 15 Hours |
| Prepare design presentation | 10 Days | 4 each | 6 | 24 Hours |
| Finalize and submit design document | 10 Days | 3 each | 6 | 16 Hours |
| Finalize roles and start implementing arch. models | 21 Days | 15 each | 6 | 90 Hours |
| Complete unit testing and begin integration testing | 28 days | 25 each | 2 | 50 Hours |
| Provide alpha version | 21 days | 10 each | 6 | 60 Hours |
| Finalize Revisions | 21 days | 15 each | 6 | 90 Hours |
| Release beta | 28 days | 15 each | 6 | 90 Hours |
| Finalize user manual/documentation | 35 days | 10 each | 2 | 20 Hours |
| Release final version | 21 days | 15 each | 6 | 90 Hours |
| Final project and presentation | 21 days | 10 each | 6 | 60 Hours |
| Demo | 7 days | 4 each | 6 | 24 Hours |

Above we can see each task broken down into a more specific estimate of hours needed to complete. We followed the class recommendation of about 30 hours a week of work while creating our iteration lengths. Each task does not require all members to participate, but in the planning phase many of them do.

For each task, not only did we estimate the amount of hours needed but also the amount of people that would participate in each task. Later in the project, tasks can more easily be broken down into front end and back end and the team members who are assigned to each domain will be more focused on those tasks.

Starting with the "Finalize roles and start implementing arch. models" task, the estimates get much broader and the tasks less decomposed. This is the beginning of next semester and thus it is harder to estimate accurately the amount of work for each task.

Our personal effort chart does not schedule out when the tasks will be completed and also does not include in class assignments that do not contribute to the project.

### 2.7 Other Resource Requirements

For our project, there are only a few resources required. Aside from developmental resources, like computers, we will need server space to host our backend and database. The space the database occupies will be fixed as initially we are only hosting the data we already have. As a stretch goal, we have the opportunity to include more data and that will contribute to the amount of space we need. Other than that, the backend isn't hosting much more than a python script that will run calculations and be available for REST queries.

We also may need to host our front end using a hoster like Vercel, AWS, or GCP. For proof of concept projects, these sites rarely need to be paid for, but there are cases where we would need to purchase hosting credits.

### 2.8 Financial Requirements

The only financial requirements needed for our project is if we need to pay to host our front end, as mentioned above. During the course of our project, it is doubtful we will hit the page enough for the sites to demand payment, however it could become necessary.

For AWS Lambda, the pricing is $0.20 for 1 million requests or $0.0000166667 for every GB-second.

There is no financial requirement for our backend as Iowa State has agreed to give us server space for free.

# 3 Design

### 3.1 Previous Work And Literature

Include relevant background/literature review for the project

– If similar products exist in the market, describe what has already been done

– If you are following previous work, cite that and discuss the **advantages/shortcomings**

– Note that while you are not expected to "compete" with other existing products / research groups, you should be able to differentiate your project from what is available

Detail any similar products or research done on this topic previously. Please cite your sources and include them in your references. All figures must be captioned and referenced in your text.

## 3.2 Design Thinking

Detail any design thinking driven design "define" aspects that shape your design. Enumerate some of the other design choices that came up in your design thinking "ideate" phase.

## 3.3 Proposed Design

Include any/all possible methods of approach to solving the problem:

- Discuss what you have done so far – what have you tried/implemented/tested?
- Some discussion of how this design satisfies the **functional and non-functional requirements** of the project.
- If any **standards** are relevant to your project (e.g. IEEE standards, NIST standards) discuss the applicability of those standards here
- This design description should be in **sufficient detail** that another team of engineers can look through it and implement it.

## 3.4 Technology Considerations

Highlight the strengths, weakness, and trade-offs made in technology available.

Discuss possible solutions and design alternatives

## 3.5 Design Analysis

- Did your proposed design from 3.3 work? Why or why not?
- What are your observations, thoughts, and ideas to modify or iterate over the design?

## 3.6 Development Process

Discuss what development process you are following with a rationale for it – Waterfall, TDD, Agile. Note that this is not necessarily only for software projects. Development processes are applicable for all design projects.

## 3.7 Design Plan

Describe a design plan with respect to use-cases within the context of requirements, modules in your design (dependency/concurrency of modules through a module diagram, interfaces, architectural overview), module constraints tied to requirements.

# 4 Testing

Testing is an **extremely** important component of most projects, whether it involves a circuit, a process, or software.

1. Define the needed types of tests (unit testing for modules, integrity testing for interfaces, user-study or acceptance testing for functional and non-functional requirements).
2. Define/identify the individual items/units and interfaces to be tested.
3. Define, design, and develop the actual test cases.
4. Determine the anticipated test results for each test case
5. Perform the actual tests.
6. Evaluate the actual test results.
7. Make the necessary changes to the product being tested
8. Perform any necessary retesting
9. Document the entire testing process and its results

Include Functional and Non-Functional Testing, Modeling and Simulations, challenges you have determined.

## 4.1 Unit Testing

– Discuss any hardware/software units being tested in isolation

## 4.2 Interface Testing

– Discuss how the composition of two or more units (interfaces) are to be tested. Enumerate all the relevant interfaces in your design.

## 4.3 Acceptance Testing

How will you demonstrate that the design requirements, both functional and non-functional are being met? How would you involve your client in the acceptance testing?

## 4.4 Results

– List and explain any and all results obtained so far during the testing phase

- Include failures and successes
- Explain what you learned and how you are planning to change the design iteratively as you progress with your project
- If you are including figures, please include captions and cite it in the text

## 5 Implementation

Describe any (preliminary) implementation plan for the next semester for your proposed design in 3.3.

## 6 Closing Material

### 6.1 Conclusion

Summarize the work you have done so far. Briefly re-iterate your goals. Then, re-iterate the best plan of action (or solution) to achieving your goals and indicate why this surpasses all other possible solutions tested.

### 6.2 References

List technical references and related work / market survey references. Do professional citation style (ex. IEEE).

### 6.3 Appendices

Any additional information that would be helpful to the evaluation of your design document.

If you have any large graphs, tables, or similar data that does not directly pertain to the problem but helps support it, include it here. This would also be a good area to include hardware/software manuals used. May include CAD files, circuit schematics, layout etc,. PCB testing issues etc., Software bugs etc.