1. **Introduction**
   1. **Purpose of the System**

Astunc is a bullet hell type of game. Bullet hell is a subgenre of shoot them up games. The main objective is to destroy the enemy entities on screen with the projectiles from user. Usually these games are set on space. We can describe this kind of game as Chicken Invaders with more projectiles on screen that limits the movement of the player. We aim this game to be easy to pick up and play but has a degree of difficulty for people to master.

* 1. **Design Goals**
     1. **Usability:** We want our game to be easy to pick up and play because of this we don’t want any of our interfaces to be too complicated so the user can find anything he/she is searching for.
     2. **Ease of Learning**: Since we want the user to learn the game quickly we didn’t put too much complicated systems or mechanics in to the gameplay and the tutorial mode is key to making the game easy to understand.
     3. **Extendibility:** To increase the longevity of the game we want to add new features such as new stages, different player ships, enemies and power ups easily without changing too much of the implementation.
     4. **Portability:** To make our game portable we will implement our system in Java to make use of the platform independency that comes with it.
     5. **Modifiability:** In order to make modifications on the existing parts of the game easy we will minimize the coupling of the subsystems as much as possible, to avoid undesired changes in other parts of the system.

1. **Software Architecture**
   1. **Overview**

This part will be about decomposing our system into subsystems which will help us to maintain the complexity much more easily. In dividing these subsystems, our main concern is about reducing the coupling between subsystems of the Astunc, on the other hand increasing the cohesion of subsystem components. In addition we tried to decompose our system in order to apply MVC( Model View controller ) architectural style on our system.

* 1. **Subsystem Decomposition**

Our system got parts that specialized to perform related actions which helps us to have better view on organization of our system. We tried to minimize coupling of the subsystems and maximize their cohesion to increase extendability and modifications that can be done when needed.

* 1. **Hardware/software Mapping**

Astunc will be implemented using Java as programming language. As hardware, Astunc requires a basic keyboard for playing the game, navigating the menus and typing. Since we will implement the project in Java system requirements will be a basic computer with operating system and a java compiler. System will not require any Internet connection to operate and there will not be any complex database.

* 1. **Architectural Style**

Since we are using MVC architectural style, our system is decomposed by three layers, User Interface, Game Data Handler, Game Data. Hierarchy between these layers is as follows. User Interface is above all since it is about displaying and interacting with the user. Game Data Handler comes second, computes the game logic and arranges the data-flow between User Interface and Game Data. Game Data is the last and holds the necessary variables and objects.

In this architectural style, the main approach is classifying the subsystems into three parts, called model, view and controller. By dividing the subsystems into three parts, we isolate the domain knowledge from the user interface by adding a controller part between them. In our system, we grouped our domain objects into game entities layer which constitutes the model of our system. The domain objects of our system is only accessed and controlled by manager classes which are grouped under Game Data Handler layer that constitutes the controller part. We grouped the classes which are responsible for providing the interaction between user and system into User Interface layer, this layer constitutes the View part since it just communicates with the model part via controller part. By this architecture it is achieved that changes on the interfaces do not change the model of the system, therefore it is a good choice to use MVC for games.

* 1. **Persistent Data Management**

Game data will be stored in the client hard disk drive, we will not use any database since the data we use in the game needs to be accessed in real-time. All the necessary files will be loaded to memory and accessed as needed.

* 1. **Access Control and Security**

Astunc will not require any kind of internet connection. And since there isn’t any kind of database to hold credentials as well as no need for any authentication anyone who runs the program will be able to play the game. And since there are no kinds of user profile there won’t be any kind of security problems

* 1. **Boundary Conditions**

Astunc will not require any kind of install. Astunc can be closed using the Quit button in the main menu if the player wants to quit during gameplay player must first pause the game then return to main menu in order to quit. When player finishes all of stages game will end and player will be sent to the display high scores screen and this screen will update to show current player’s name if a record is broken. Game will give an error if the file is corrupted.

1. **Subsystem Services**