**Remote Mental Health Management**

**“Naaman”**

Architecture Notebook

There is guidance within this template that appears in a style named InfoBlue. This style has a hidden font attribute that allows you to toggle whether it is visible or hidden in this template. Use the Microsoft® Word® menu **Tools > Options > View > Hidden Text** check box to toggle this setting. There is also an option for printing: **Tools > Options > Print**.

# Purpose

The purpose of this document is to describe in detail the architectural goals and philosophy, assumptions and dependencies, architectural requirements, decisions, constraints, architectural mechanisms, key abstractions, architectural framework, and views that will act as a blueprint for the design and implementation of project “Naaman”.

# Architectural goals and philosophy

Project “Naaman” is about creating a safe and comforting social media environment for mental health management. It is a platform where people are able to share their stories, struggles or what they are going through and get professional help from trained mental health professionals and medical professionals.

Due to the system being heavily regarding mental health management, it has to be reliable, with no server downtime, user friendly with good navigation design, and have good performance with fast loading times. The user interface has to be easily understandable with no corner page. A backend interface is necessary to provide the backend programmers easy access to do updates and maintenance.

* **Reliable** - because the system is about health management, it must be reliable. People will be using the system to share personal information and contact mental health professionals or medical professionals, because of this function the backend server has to be reliable to avoid losing user personal data, appointment schedule information. In cases of natural causes out of our control such as disastrous weather conditions, there must be back up generators to keep the backend server running during such events.
* **User friendly** - the user interface must be easy to understand and give a welcoming feeling to users. It should not look rigid and complicated. Users should be able to adjust to using the system quickly, not get confused and stressed at first glance. The system should have good navigation, where users are able to understand where to go easily depending on what they are looking for. Such as a big drop down menu for sharing stories or making appointments. Lastly, there must not be corner pages, which are pages where there is no “back” navigation. All interfaces should have an option to go back, or an option to go to the main page. Users should not feel stuck while using the system.

**Fast performance** - the system should have fast performance. It should have fast load times and fast response times. Users should not be stuck waiting for the system to load. It should run smoothly and be dynamic.

# Assumptions and dependencies

|  |  |
| --- | --- |
| **Assumption and Dependencies** | **Descriptions** |
| Internet | Naaman is a web-based project, therefore it is necessary to obtain a domain name, as well as web hosting services. |
| Client base | Our main client base are people experiencing difficult times, having mental challenges, stress, loneliness etc. |
| Server | A server is necessary to keep the system running. |
| Database server | A database server is necessary to document and maintain sensitive user information. |
| **Project team** | **Contact information** |
| Sobana Handi Achini Thisarangi De Silva | achinithisarangi86@gmail.com |
| Dehemi Vihara Dissanayake Liyanage | dissanayake7793@gmail.com |
| Gaury Chethana Thanthirigama | chethana9804@gmail.com |
| Temuulen Tsengel | temuulen2830@gmail.com |
| Michael Weisang | weisangm1@gmail.com |

# Architecturally significant requirements

“Naaman” will be a dynamic website that will have a reliable server with backup generators in case of power outages or natural disasters, user friendly and robust user interface with fast response and load times.

A backend interface is necessary to ensure server maintenance is up to date. Programmers working on the website after development finishes need an interface to make future maintenance and updates of the website efficient.

Database server is essential to maintain and document essentially everything that happens on the website. Including user information, appointments or booking details, and version control of regulations.

# Architectural Mechanisms, Constraints, and Justifications

|  |  |  |
| --- | --- | --- |
| **Architectural Mechanisms** | **Constraints** | **Justifications** |
| Reliability | Natural causes (weather disasters) | We need the system to be reliable to ensure no loss of data occurs. |
| User Friendliness | Clustered or confusing interface design | We need a user friendly user interface to allow users to easily use the system |
| Fast Performance | Internet speed | We need fast performance to avoid user frustration over slow loading times |

# Key abstractions

* User - People who are going through stressful times or experiencing difficulties
* Remote Mental Health Management system - User interface of website “Naaman”
* Server - Backend server which runs the website
* Database - Server to maintain and document user sensitive data
* Backend interface - Interface for programmers to update and maintain the system
* Mental health professionals and medical professionals

# Layers or architectural framework

This system will be developed using the three-tier architecture, which are client layer, server layer, database layer. By using the three-tier architecture we ensure each aspect of the system is addressed and covered in development.

**Client layer** - covers user interface. The user interface should be user friendly with good navigation. Client layer covers all aspects of user related decisions.

**Server layer** - covers server requirements.The server must be reliable to ensure the system is always available. In cases of unpredictable disastrous situations, a backup generator must be there to keep the server running.

**Database layer** - covers database server. The database server will contain the client’s sensitive information, it must have good security. The backend interface should cover all aspects of updating and maintaining the system.

# Architectural views

* Account creation - Users are able to create an account. Once they do they have the option to decide the privacy level of stories they share.
* Sharing stories - Users are able to read and comment on other stories.
* Comments - Users are able to give points or reduce points on comments depending on how helpful it was or not
* Online appointments - Users are able to make appointments with mental health professionals
* Access to professional help - Users have access to professional help in many ways (professional advice posts or booking a session with professionals).
* Support based on types - Users have the option to choose what kind of help they need based on what they are going through (symptoms).
* Peer reviewed articles - Professional articles are available.