

**Inward Module (HIS)**

**Application Frameworks (SE3040)**

**Technical Report**

**3rd Year 1st Semester**

**WE05 - Axiom**

|  |  |
| --- | --- |
| **M.R. Aaquiff Ahnaff** | **IT15048738** |
| **T.T.T. Rathnaweera** | **IT15011930** |
| **H.P.R. Lakshan Thilakarathne** | **IT15078360** |
| **Bhanuka Hingalagoda** | **IT15013224** |

# **Abstract**

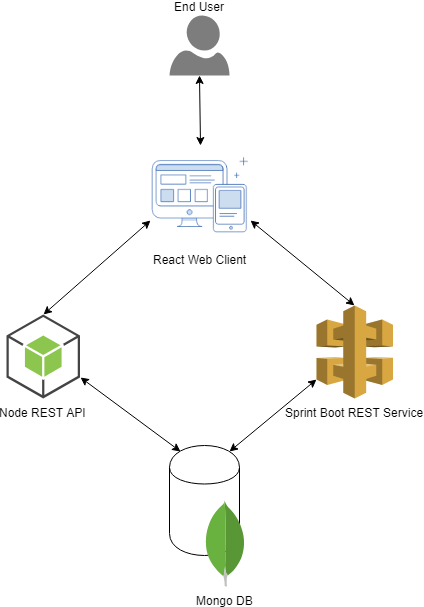
This report discusses the implementation of a health care software system to be used in hospitals to manage the Inward related functions. This project was carried out as part of a requirement to complete the 3rd year of B.Sc. (Hons) Degree in Information and Technology Specialization in Software Engineering at Sri Lanka Institute of Information and Technology.

# Introduction

The Inward system is designed to cater the requirements for managing wards in a hospital. It will be part of a suite of hospital management software suite called HIS. This is a full stack application built using ReactJS, NodeJS and Mongoose. The application would be able to manage wards, beds, patient admissions, discharge, archive details, patient prescriptions, laboratory tests, patient allergies, patient transfers and generate charts and reports.

# Design

This full stack application is designed to be easily decoupled thus allowing the swapping of the client or server modularly.



The application consists of three layers,

* Client
* Server
* Database

The client is developed using ReactJS which is an open source JavaScript library that helps build web clients.

The server is built on NodeJS which is a framework for building web applications and combined with express framework for node, we can build restful web services.

The database is a NOSQL database based on mongoose and hosted in a cluster on MongoDB Atlas Cloud storage. To connect the database to the NodeJS application we used the mongoose driver.

# Implementation

## Authentication

Authentication of the application is done using web tokens. JWT is a library that can be used to generate and verify the tokens and is used in the NodeJS server. When the user logins, a token is generated and sent back to the client in the response, the client has to use this token to access the protected routes which are followed by /api in the url.

The react client uses universal-cookies library to store the token in the web browsers local storage as cookies when the user logs in. For all the subsequent request for the protected resources it will fetch the token from cookie and include it as a authentication header for those requests.

## Rest API calls

The react application uses the library ‘axios’ to make REST calls to the server. Axios is a promise based http client for the browser.

## Promise Based DAO

The NodeJS project uses promises for IO operations like database access to MongoDB. Promises represent the success or failure of an asynchronous operations through two methods resolve and reject.

## Bootstrap CSS

For styling of HTML content, we will use Bootstrap 4 as the styling sheet. Bootstrap is an open source toolkit for developing HTML, CSS and JS.

# Deployment

As discussed before the database is hosted in Mongo Atlas cloud. The client and the server are hosted in an Amazon EC2 server provided by cloud provider Amazon Web Service(AWS).

The EC2 instance will fetch the source code for the projects from GitHub using git for Linux. Then each of these projects are run in the background using the command “npm start &”. To run this command, we need to log into the EC2 instance using SSH.

To allow traffic to the EC2 server it needs to be assigned a security group with rules allowing the traffic on port 8080 which exposes the react client and also port 3000 which exposes the NodeJS rest API which can be used by external clients.

# Repository

The repositories for the source code are hosted on GitHub and given below are the links to these repositories

* React Client - <https://github.com/Aaquiff/AF-React>
* Node Server - <https://github.com/Aaquiff/AF-NodeJS>