# **Notes Web Application**

# Architecture/Design Document

# **Table of Contents**

[1 Introduction 3](#__RefHeading___Toc131876418)

[2 Design Goals](#__RefHeading___Toc131876419) 3

[3 System Behavior](#__RefHeading___Toc131876420) 3

[4 Logical View](#__RefHeading___Toc131876421) 3

[4.1 High-Level Design (Architecture)](#__RefHeading___Toc131876422) 3

[4.2 Detailed Module Design](#__RefHeading___Toc131876425) 4

[4.3 Detailed Database Design](#__RefHeading___Toc131876425) 4

[5 Use Case View](#__RefHeading___Toc131876429) 5

Change History

**Version:** v1.0

**Modifier:** Yugandhar Gangu

**Date:** 2016/11/19

**Description of Change:** Initial creation

# Introduction

This document describes the architecture and design for the Notes Web Application being developed for the Infoteria Corporation. Notes Web Application a web application that manages a list of notes. Pocket Campus Tour has a roaming mode that requires zero computer skills. Users can create their account by sign up and sign in to application with successful registration. User can manage his/her notes by create, view, edit and delete.

# Design Goals

The priorities for the design that follows are:

* The design should minimize complexity and development effort.

# System Behavior

The use case view is used to both drive the design phase and validate the output of the design phase. The architecture description presented here starts with a review of the expect system behavior in order to set the stage for the architecture description that follows. For a more detailed account of software requirements, see the requirements document.

**Client**

Web Browser

**Node Server**

Node App

**Database**

Postgre SQl

Figure 1 System Behavior

# Logical View

## High-Level Design (Architecture)

The high-level view or architecture consists of 3 major components:

* The **Web Browser** is to send the request and display the server response.
* The **Node Server** is a server that takes request from web browser and handles it. It may communicate with Database in case of saving or fetching the data.
* The **Database** is to save the user or notes information.

**Web Browser**

HTML

Angular Js

**Node Server**

Node Js

Express Js

Sequelize Js

**Database**

Postgre Sql

Figure 1 System Architecture

## Detailed Module Design

**Routes**

Index.js

Notes.js

**Root**

app.js

package.json

**Bin**

www

**Views**

changepassword.jade

error.jade

index\_layout.jade

index.jade

login.jade

noteinfo.jade

notes\_layout.jade

notes.jade

signup.jade

**Models**

users.js

notes.js

## Detailed Database Design

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **m\_users** | | | | | | | |
| **Column Name** | **Datatype** | **Length** | **NULL** | **Default** | **PK** | **FK** | **Details** |
| id | int | 11 | N |  | Y |  |  |
| email\_id | varchar | 100 | N |  |  |  |  |
| full\_name | varchar | 100 | N |  |  |  |  |
| last\_login\_at | timestamp |  | Y |  |  |  |  |
| login\_count | int | 11 | N | 0 |  |  |  |
| active\_flag | byte | 1 | N | 1 |  |  |  |
| modify\_at | timestamp |  | Y |  |  |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **m\_auth** | | | | | | | |
| **Column Name** | **Datatype** | **Length** | **NULL** | **Default** | **PK** | **FK** | **Details** |
| id | int | 11 | N |  | Y |  |  |
| user\_id | int | 11 | N |  |  | Y | m\_users |
| password | varchar | 250 | N |  |  |  |  |
| active\_flag | byte | 1 | N | 1 |  |  |  |
| created\_at | timestamp |  | N | NOW |  |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **m\_note\_subject** | | | | | | | |
| **Column Name** | **Datatype** | **Length** | **NULL** | **Default** | **PK** | **FK** | **Details** |
| id | int | 11 | N |  | Y |  |  |
| user\_id | int | 11 | N |  |  | Y | m\_users |
| subject | varchar | 250 | N |  |  |  |  |
| active\_flag | byte | 1 | N | 1 |  |  |  |
| created\_at | timestamp |  | N | NOW |  |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **m\_notes** | | | | | | | |
| **Column Name** | **Datatype** | **Length** | **NULL** | **Default** | **PK** | **FK** | **Details** |
| id | int | 11 | N |  | Y |  |  |
| note\_id | int | 11 | N |  |  | Y | m\_note\_subject |
| notes | varchar | 1000 | N |  |  |  |  |
| version\_no | int | 5 | N | 1 |  |  |  |
| active\_flag | byte | 1 | N | 1 |  |  |  |
| created\_at | timestamp |  | N | NOW |  |  |  |

# Use Case View

**5.1 Sign up:**

Route: index.js

View: signup.jade

Model. user.js

Database: m\_users, m\_auth

1. Start.
2. Get request from client (Web Browsr).
3. Start Get method in index.js route.
4. Render signup.jade.
5. Return to Client.
6. Get the below inputs from client through AngularJS.
7. Validate inputs. If valid inputs send to server, Otherwise prompt related error message.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input Name** | **Field Name** | **Max Length** | **Mandatory** | **Validation Message(s)** |
| Email Id | email\_id | 100 | Yes | Please input email address.  Please input valid email address. |
| Full name | full\_name | 100 | Yes | Please input full name. |
| Password | password | 25 | Yes | Please input password. |
| Confirm Password | confirm\_password | 25 | Yes | Please input password.  Password and Confirm Password is not matching. |

1. Get the inputs from Client in index.js route.
2. Handover email\_id input to users.js model checkExists() method. If checkExists() method returns true then send error message to client. Otherwise Handover inputs to users.js saveUser() method.

Error Message:

{

“code”:”1”,

“message”:”The Email Id is already registered. Please go to sign in.”

}

saveUser() Mapping:

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Table Name** | **Column Name** | **Value** |
|  | m\_users | id | Auto Generate |
| email\_id | m\_users | email\_id |  |
| full\_name | m\_users | full\_name |  |
|  | m\_auth | id | m\_user → id |
| password | m\_auth | password | MD5 encrypted |

1. Get the user\_id from users.js model saveUser() method after successful registration.

Success Message:

{

“code”:”0”,

“message”:”Registration is successful. Please sign in to continue.”,

“user\_id”:”X”

}

Note:

X – User Id (Auto generated by postgreSQL)

1. Send response to client. (Error Message or Success Message)
2. AngularJs signup ajax call will catch the response shows in web browser.

Error Response:

a. Show error message.

Success Response:

a. Show success message.

b. Sign in button. (OnClick – Redirect to Sign in screen)

1. End.

**5.2 Sign in:**

Route: index.js

View: signin.jade

Model. User.js

Database: m\_users, m\_auth

1. Start.
2. Get request from client (Web Browsr).
3. Start Get method in index.js route.
4. Render signin.js.
5. Return to Client.
6. Get the below inputs from client through AngularJS.
7. Validate inputs. If valid inputs send to server, Otherwise prompt related error message.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input Name** | **Field Name** | **Max Length** | **Mandatory** | **Validation Message(s)** |
| Email Id | email\_id | 100 | Yes | Please input email address.  Please input valid email address. |
| Password | password | 25 | Yes | Please input password. |

1. Get the inputs from Client in index.js route.
2. Handover email\_id input to users.js model validateUser() method. If validateUser() method returns 0 then send error message to client. Otherwise add return value (user\_id) into session scope and send success message to user.

Error Message:

{

“code”:”1”,

“message”:”Invalid credentials. Please try again.”

}

Success Message:

{

“code”:”0”,

“message”:”Sign in successful.”,

}

Note:

X – User Id (Fetch from postgreSQL)

1. Send response to client. (Error Message or Success Message)
2. AngularJs signin ajax call will catch the response shows in web browser.

Error Response:

a. Show error message.

Success Response:

a. Show success message.

b. Redirect to notes.js module.

1. End.

**5.3 Add/View/Edit Note:**

Route: notes.js

View: noteinfo.jade

Model. notes.js

Database: m\_note\_subject, m\_notes

1. Start.
2. Get request from client (Web Browsr).
3. Start Get method in index.js route and fetch note id from request.
4. Pass note is to notes.js model and fetch note data in json format.

Format:

{

“id”:”X”,

“subject”:”xxxxxxxx”,

“note”:”xxxxxxxx”,

“action”:”Y”

}

X – note id

xxxxxxx – content

Y – action type (0:view, 1:add, 2:edit)

1. Render noteinfo.jade view.
2. Return to Client.
3. Get the below inputs from client through AngularJS.
4. Validate inputs. If valid inputs send to server, Otherwise prompt related error message.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input Name** | **Field Name** | **Max Length** | **Mandatory** | **Validation Message(s)** |
| Subject | subject | 100 | Yes | Please input subject. |
| Note | note | 1000 | Yes | Please input note. |

1. Get the inputs from Client in notes.js route.
2. Handover inputs to notes.js model save() method.

Message:

{

“code”:”0”,

“note\_id”:”X”

“message”:”Note has been Saved successfully.”,

}

Note:

X – Note Id (Fetch from postgreSQL)

1. Send response to client. (Success Message)
2. AngularJs signin ajax call will catch the response shows in web browser.

Success Response:

a. Show success message.

b. Redirect to notes.js module.

**5.4 Note List:**

Route: notes.js

View: notes.jade

Model. Users.ja, notes.js

Database: m\_note\_subject, m\_notes

1. Start
2. Get request from client (Web Browser).
3. Start Get method in index.js route.
4. Fetch user\_id from session and pass it to notes.js model and fetch full name and last login time from m\_user table.
5. Render notes.jade view.
6. Return to Client.
7. AngularJs will fetch list of notes through ajax call.
8. Ajax request will send to server.

Request format:

{

“page\_no”:”X”,

“rows\_per\_page”:”X”

}

X → integer

1. Get ajax request at server side in notes.js route.
2. Fetch list of notes from m\_note\_subject and m\_notes table.
3. Send REST response to client.

Response Format:

{

“code”:”X”,

“count”:”X”,

“notes”:[

{

“note\_id”:”X”,

“subject”:”xxxxxxxx”,

“note”:”xxxxxxxx”,

“date”:”xxxx-xx-xx”,

“time”:”xx:xx xx”,

}

]

}

1. AngularJS ajax call will show notes to user in format.
2. Below components will display along with notes.

Checkbox in each line → to select multiple checkboxes.

Delete button → notes delete request will send to server.

Edit button → Redirect to edit note page

View button → Redirect to view note page

1. End.

**5.5 Delete Note:**

Route: notes.js

Model: notes.js

Database: m\_note\_subject

1. Start
2. Get request from client (Web Browser).
3. Start Get method in notes.js route.
4. Fetch note\_id from request and pass it to notes.js model deleteNote() method. DeleteNote() method will update active\_flag to 0 in m\_note\_subject table.

Request Format:

{

“note\_id”:”X”

}

1. Return to Client with success message.
2. AngularJs will fetch success message through ajax call and shows success message to client.

Response Format:

{

“code”:”X”,

“message”:”xxxxxx”

}

1. Update note list through call.
2. End

**5.6 Delete Note:**

Route: notes.js

View: changepassword.jade

Model: users.js

Database: m\_users, m\_auth

1. Start.
2. Get request from client (Web Browser).
3. Start Get method in notes.js route.
4. Render changepassword.jade view.
5. Return to client.
6. AngulatJS will check users inputs. If valid send to server, Otherwise show prompt message in client.
7. Fetch user\_id from session and pass it to notes.js model updatePassword() method. The updatePassword() method will update all of the records of current users m\_auth table's active\_flag to 0. Insert new record to m\_auth table.

Request Format:

{

“old\_password”:”xxxxxxxx”,

“new\_password”:”xxxxxxxx”

}

1. Return to Client with success message.
2. AngularJs will fetch success message through ajax call and shows success message to client.

Response Format:

{

“code”:”X”,

“message”:”xxxxxx”

}

1. End