|  |  |  |  |
| --- | --- | --- | --- |
| Application : My School | Release Version: 1.0 | Document Revision: 0.2 |  |
| Last updated date : 14 Oct, 2016 | | |
| Author : Zakir Ahmed | | |
| My School – Requirement Specification | | | |
| Company and brief information :  Accolite is one of the fast growing companies in the IT consulting space using the latest cutting edge technologies. | | | Company Links and references : |

**DOCUMENT HISTORY**

* REVISION HISTORY

The list of revisions of the release document along with summary of changes delivered.

|  |  |  |  |
| --- | --- | --- | --- |
| Revision Number | Revision Date | Summary of Changes | Author |
| 0.1 | 10-10-2016 | Initial Version | Zakir Ahmed |
| 0.2 | 14-10-2016 | Added SSL verification for STAFF webservice |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

* APPROVALS

The release document has been approved by the below approvers.

|  |  |
| --- | --- |
| Name | Title |
| Dileep Kumar Kamujula | Project Lead |
|  |  |
|  |  |
|  |  |
|  |  |

* DISTRIBUTION

The list of people/groups to whom the document needs to be distributed.

|  |  |
| --- | --- |
| Name | Title |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**ABOUT THE DOCUMENT**

* ABOUT THE DOCUMENT

This document deals with the requirement specification of web application “My School”. It details the features that would be delivered as part of release 1.0

* WHO CAN USE IT
* Developers
* Project Managers.
* CONVENTIONS USED:

This section lists all the conventions used in the document for better understanding.

* Code and code fragments and properties shall be shown in the format code . If any code needs to be highlighted, it is highlighted in orange colour.
* Flower brackets {} surround shall be used for dynamic values for logical explanation, but not for code.

1. **OVERVIEW**

The application “My School”, a single solution to manage all the activities/processes related to running a school, being it managing the students, staff, transport, attendance, fees, payments etc., Please find the list of modules currently covered in release 1.0

* Students
* Classes
* Staff
* Transport
* Fee management
* Configuration

The features which shall be planned for future releases are

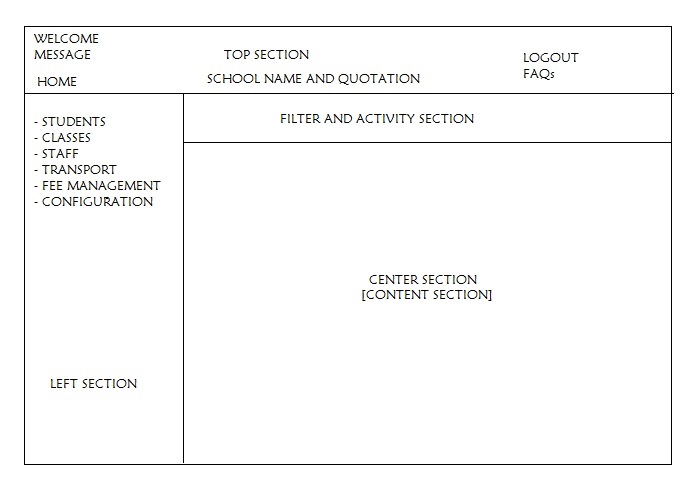
* Examination and Evaluation records
* Attendance
* Payments
* Reports
* Events

We shall get into the release 1.0 features in detail in the next sections.

1. **BASIC LAYOUT**

The layout of the site comprises of three major sections as outlined below.

* Top Section
* Left Section (which includes options for navigating the site)
* Center Section (a Partial View Loaded by Ajax. Either the data or the whole HTML based on the use case, which displays the details of the option selected)
* All the data layer should be exposed from a Rest ful api layer and the services should not rely on User Session.
* All the APIs should have CORS enabled for a list of White Listed domains. (Should be able to read the webservice data from a different whitelisted domains without any issues.)



1. **FEATURES**
   1. **LOGIN**

The initial login page shall have three basic input details.

* Username
* Password
* Institution

We shall store the list of valid users (take only admin roles for now) for each school and shall store in database. If all three parameters match, we shall allow the user to login, otherwise a proper error message shall be returned.

* 1. **STUDENTS**

By clicking the Students link in the options section, the center section will load the list of students. The layout of the center section would have

* Filter section: The student list filter based on name and the class, along with apply filter button. It would also have a button to register a new student.
* Content section: The content section will load the list of students based on the filter applied. Initially when the page is loaded, there will be no filter applied and the students are listed in alphabetical order by default.

**New registration**: When we click on New Registration button, a form is opened in the content section, which would capture the below details of the student. The form should have a register and clear buttons. When the clear button is clicked, clear the complete form. When submit button is clicked, the student should be registered and saved in the school database. Based on the extra facilities opted by the student, fee liabilities would be calculated at saved. When a new student is registered, an email/sms should be triggered to the number and email address provided.

* Student Name
* Student Class
* Joining Date
* Last updated time
* Address
* Fathers name
* Mothers name
* Contact number/Alternate contact number
* Email address
* A check-list of extra-curricular activities student wants to opt for.

**Students List**: For each student record displayed, there should be an edit/remove button. When edit button is clicked, the details of the student should be displayed in edit mode, the details can be changed and saved. When we click the remove button, there will be a confirmation popup the student shall be marked as inactive. [To keep it simple, we want the user to check all the pre-requisites manually before removing a student like clearing dues like fees, books, other school items. We can automate later]. Whenever a student is edited/removed an email/sms shall be triggered to the number and the email address provided. Paging functionality should be added to the list of students by displaying 20-25 students in a single view.

* 1. **CLASSES**

By clicking the Classes link, the center section will load the list of classes available in school. The layout of the center section would have

* Filter section: Empty/invisible
* Content section: The content section will load the classes based on the school configuration.

**Class List**: For each class record displayed, there should be a schedule button. When schedule button is clicked, the time table for the class would be rendered. All this data shall be picked from the school configuration. It is not possible to edit the schedule and add/edit/remove the classes from this page.

**Use a Rest Service to fetch the data in JSON format and display the result on UI by parsing it in JS**.

* 1. **STAFF**

By clicking the staff link, the center section would load all the staff of the school including the principal. The layout of the center section would have

* Filter section: A name and staff type filters (house-keeping, administration, security, teacher etc). There will be a button at the top to add a new staff and a button to list the open staffing requests and their status. There shall also be buttons to assign/un-assign staff to a class.

**Note :** The API should implement Service side filtering using Lambda and Stream APIs.

* Content section: The content section will load the active staff list based on the filter selected.

**New staff**: When we click on New staff button, a form is opened in the content section, which would capture the below details of the staff requirement. The form should have a submit and clear buttons. When the clear button is clicked, clear the complete form. When submit button is clicked, there would be a REST request would be fired to the STAFF web service and get an immediate acknowledgement if the request is accepted or not. The STAFF app would send updates as an async response with the status of the request and when the candidate is available it will send an async response with the details of the new STAFF. Once we approve and get a positive acknowledgement, we shall store the details of that staff.

* Staff type
* Gender
* Minimum Experience
* Age restrictions
* Max salary
* Latest joining date.
* Class (Optional)
* Subject (Optional)

Note : Use Kafka to notify all the other Faculty members on the portal about any new staff member joining with. Based on the person’s profile about receiving updates, send out emails to the users as and when a new Faculty member joins.

Ensure that the STAFF webservice is SSL enabled. Generate a certificate using jdk tools and save in the key store of STAFF webservice. Provide the certificate to the client to store in trusted certificates.

**Staffing request List:** On clicking the staffing requests button, all the non-closed staffing requests will be displayed along with their status. The status is updated every time we receive the update for that request from Staff web service. Against each request, we shall have cancel button till the staff is selected by the staff tool and sent the details. Once the staff details are shared, we shall display confirm button. Once we click the confirm button, a form shall be displayed with the list of the staff provided and option to reject/approve. Based on the option selected a request shall be sent to the STAFF web service. If we approve, the staff is registered with the school and shall be part of the staff list displayed.

**Staff List**: For each active staff record displayed, there should be a remove button. When we click the remove button, a form is opened to enter the last working day and reason for termination and there will be a request triggered to STAFF web service to end the contract and there would be an acknowledgement. Later STAFF tool would send an async approval to relieve the staff from duties and the staff record is marked as inactive. Paging functionality should be added to the list of staff by displaying 20-25 staff in a single view.

**Staff:** For each active staff record displayed, there should be a view button. When the view button is clicked, it should display all the classes assigned to the staff and the details of the staff. For each record displayed here, there shall be a un-assign button to un-assign the staff to the class.

**Assign Staff:** When you click on Assign staff button, it opens a form with the staff details and field to select the class. By clicking the assign button, staff shall be assigned to the class.

* 1. **TRANSPORT**

By clicking the transport link, the center section would load all the transports/vehicles of the school used for commuting the students and staff. The layout of the center section would have

* Filter section: A name and transport number filters. There will be a button at the top to add a new transport and a button to assign a transport to the staff or a student. There will also be a button to check the open transport requests and their status.
* Content section: The content section will load the active transport list based on the filter selected.

**New transport**: When we click on New transport button, a form is opened in the content section, which would capture the below details of the transport requirement. The form should have a submit and clear buttons. When the clear button is clicked, clear the complete form. When submit button is clicked, there would be a SOAP request would be fired to the TRANSPORT web service and get an immediate acknowledgement if the request is accepted or not. The TRANSPORT app would send updates as an async response with the status of the request and when the transport is available it will send an async response with the details of the new transport. Once we approve and get a positive acknowledgement, we shall store the details of that transport. The

* Transport type
* Minimum seats
* Earliest manufacturing year
* Max vehicle contract
* Latest rental date.

**Transport request List:** On clicking the transport requests button, all the non-closed transport requests will be displayed along with their status. The status is updated every time we receive the update for that request from transport web service. Against each request, we shall have cancel button till the transport is selected by the transport tool and sent the details. Once the transport details are shared, we shall display confirm button. Once we click the confirm button, a form shall be displayed with the list of the transports provided and option to reject/approve. Based on the option selected a request shall be sent to the TRANSPORT web service. If we approve, the transport is registered with the school and shall be part of the transport list displayed.

**Transport List**: For each active transport record displayed, there should be a remove button. Also display the number of free seats available for each transport. When we click the remove button, a form is opened to enter the last renting day and reason for termination and there will be a request triggered to TRANSPORT web service to end the contract and there would be an acknowledgement. Later TRANSPORT tool would send an async approval to relieve the transport from duties and the transport record is marked as inactive. A transport can be removed only if there are no assigned commuters to it.

**Transport:** For each active transport record displayed, there should be a view button. When the view button is clicked, it should display all the staff and students commuting by that transport. For each record displayed here, there shall be a un-assign button to free the seat. Clearance of dues to be checked manually for now and can be automated later. Upon un-assigning the transport, an email/sms notification shall be sent to the contact number and email address registered.

**Assign Transport:** When you click on Assign transport button, it opens a form with the list of available seats and field to select the student. By clicking the assign button, student shall be assigned to the transport and a fee liability shall be added for that student. Upon assigning the transport, an email/sms notification shall be sent to the contact number and email address registered.

* 1. **FEE MANAGEMENT**

By clicking the fee link, the center section would load all the pending fee liabilities of the students. The layout of the center section would have

* Filter section: A name and class filter.
* Content section: The content section will load the pending fee liability list based on the filter selected.

**Fee Liability:** When the student is registered, based on the options/facilities selected, the fee liability template is populated. The template could have a one-time admission fee, a monthly/quarterly tuition fee, a monthly/quarterly sports club fee, an annual library membership etc.

**Fee Reminders:** Based on the template, a scheduled job runs to create the liabilities out of the template and send sms/email reminders/notifications. The reminder date is calculated based on the reminder dates configured in school configuration and based on the frequency. If the fee liability is monthly in the template, then as per the configuration the deadline would be 5th of each month and the reminder could be triggered 3 days before the deadline.

Note : The configuration of the templates and job schedules should be dynamic. Use Quartz Scheduler to schedule jobs.

**Fee payments**: For each of the fee liability displayed in the list, there would be pay button. On clicking the pay button, a form is opened with the details of the fee liability and a field to accept the amount of fee paid. The payment is manual for now. The user can enter the full amount or the partial amount and click on submit. The fee payment will be saved and if it is a partial payment, the liability record would still be visible with the updated balance. Upon the payment, a sms/email notification shall be sent to contact number and email address registered.

* 1. **CONFIGURATION**

By clicking the configuration link, 4 sub-options shall be visible in the left section. Based on the sub-option selected, the center section would load the configuration accordingly. The configuration sub-options shall include the below details for now.

* **Classes:**

The list of the classes that are provided by the school. There will be buttons to add/remove a class.

* **Time table:**

There shall a filter on the top to select the class. Based on the filter, the time table for the class shall be displayed as records in chronological order. Against each record, there shall be a remove button by clicking on which the record is removed. There shall be a button to add a record, by clicking which shall open a new form with below details. Upon clicking save, the details are validated and the time table is updated.

* + Start time
  + End time
  + Subject
  + Teacher
* **Fees:**

By clicking the fee option, the list of fee liability templates would be displayed. A new liability can be added and existing ones can be edited.

1. **TECHNOLOGIES INVOLVED**

* Spring IOC, Beans
* Spring AOP (For logging)
* Spring MVC (For request routing, model and view decoupling and association)
* Spring ORM
* Hibernate/JDO
* Active MQ (for asynchronous messaging. Alternate for Kafka)
* Apache Kafka (for asynchronous messaging)
* Spring Boot (To create a rest ful STAFF web service)
* Quartz scheduler/OS level cron job (To monitor fee liabilities and send reminders)
* Spring Transaction API (for global transactions)
* Apache Velocity (For mail/sms templates)
* Apache Axis 2 (For SOAP Transport service)
* XSLT templates and XML parsers (To build a utility to view school configuration as an HTML. Later we can add functionality to export the school configuration as an xml and provide import functionality)
* JSON parsers (for STAFF client)
* Use Java7/8 enhancements as applicable.
* IText/Apache POI for reports.HTTPS certificates