Work-Flow Automation

Software Requirements Specification

Saurabh Batra
140101066

Sagar Manchanda
140101060

Siddharth Khabia
140101071

Jagjot Singh
140101029

Table of Contents

1.0 Introduction

1.1 Purpose

The purpose of this document is to present a detailed description of the Work Flow Automation API designed by Team 13 as part of CS243 (2016) course under Dr. Saurabh Joshi. It will explain the purpose and features of the API, what the API will do and the constraints under which Client and User (definitions cleared in glossary) must operate. This document is intended for the invigilators of the project as well as further developers.

1.2 Scope of the Project

As the name suggests, this API is for Automation of Overall Workflow in Institutes and organizations like ours (IIT-Guwahati). This API is designed to maximize the Coder's (check glossary) productivity by providing tools to assist in automating the process of creating all the forms, requests with or without constraints, required for administration in the Institute, which would otherwise have to be performed individually from scratch or by physical means, where the client needs to submit a form to required authority in person. The API will provide features to do this exhaustively while still remaining easy enough to work with.

More specifically, this API is designed to allow the Coder to create and publish multiple WebApps, connecting to user specific databases with unique constraints. The API will facilitate communication between Clients with varying hierarchy.

1.3 Overview of Document

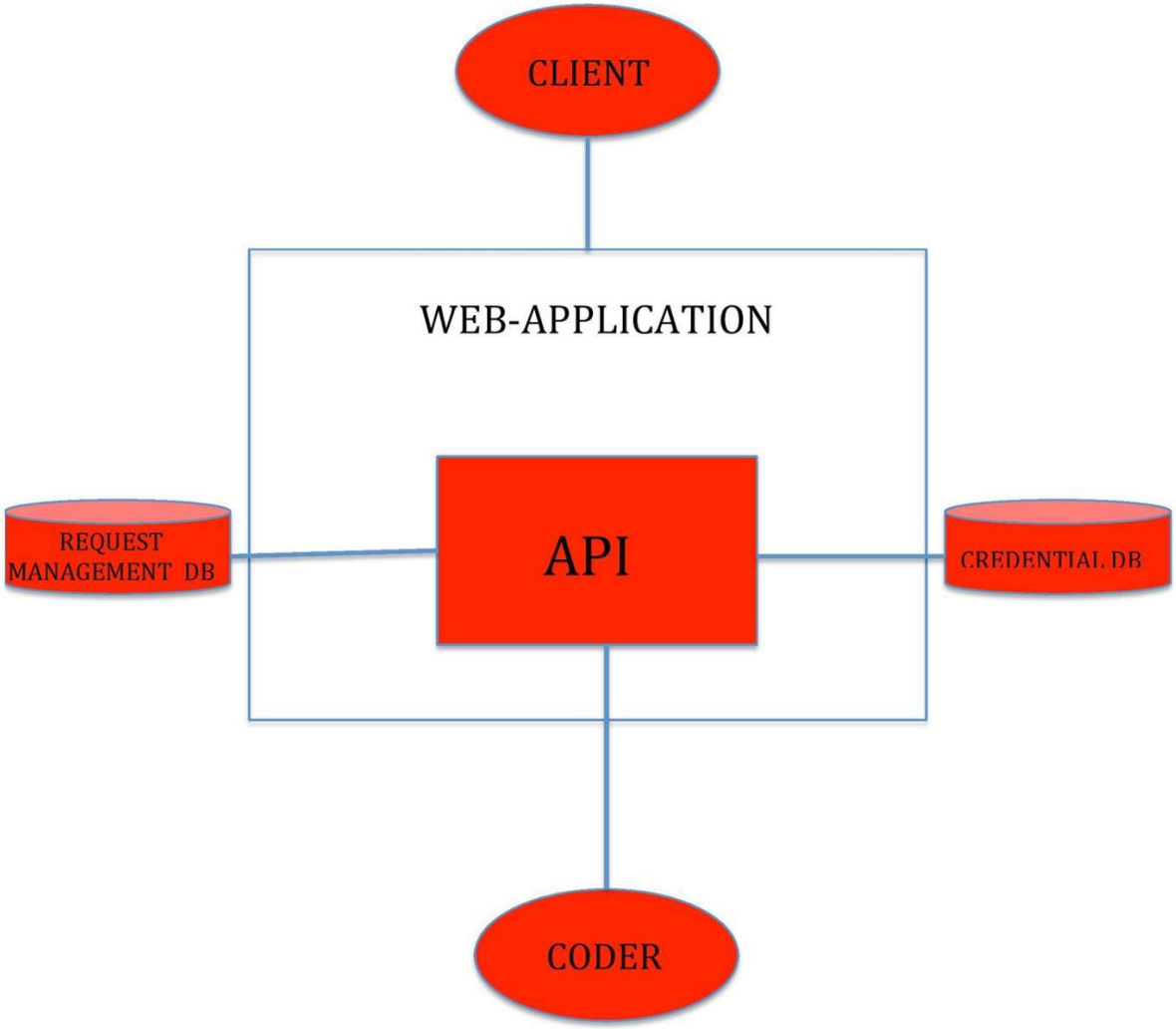
The subsequent part, the Functional Requirement Specification section, of this document gives an overview of the functionality of the API. It describes the different kind of users and developers involved and their roles.

The next part, Non-Functional Requirement Specification section of this document is written primarily for the developers and describes in technical terms the details of the non-functional aspects of the product. The last part deals with User Characteristics.

All the sections of the document describe the same software product in its entirety, but are intended for different audiences and thus use different languages.

1.4 Glossary

TERM	DESCRIPTION
Coder	The person who uses the API codebase to build WebApp on top of it.
Client	The end user, who will mske use of the Webapp to streamline and automate the workflow process.
API	The actual code, written by the developers (us), which will help ease the Coder in making the WebApp; presented in the form of an interface.
WebApp	The product obtained, involving forms and hierarchies, created by the Coder and used by the Client.
Request	A Request is submitted by the Client by filling out a form and is logged into the client database.
Form Builder	The main part of the API, involves adding the GUI form elements and handling their validation.
Hierarchy	Requests have to be approved by different clients belonging to different groups. These proceed in a hierarchy set by the Coder.



WEB FRAMEWORK FOR WORKFLOW AUTOMATION

2.0 Functional Requirements Specification

2.1 System Environment

The WFA API has three active actors, namely the developer, the coder and the client.

The developers who created the system work on improving its functionality and scope, so as to allow the Coder to create better WebApps more easily.

The Coder is the person actually using the product (WFA API), and creating constraint based WebApps with databases. These WebApps are the end product, to be used by the clients, created so as to automate the workflow of the clients.

The Clients are the end users, for whose benefit the project had been contrived. The client may have multiple hierarchy or roles in the WebApp (like an applicant and a respondent, a library member and the librarian etc.). The WebApp manages and automates the work of clients.

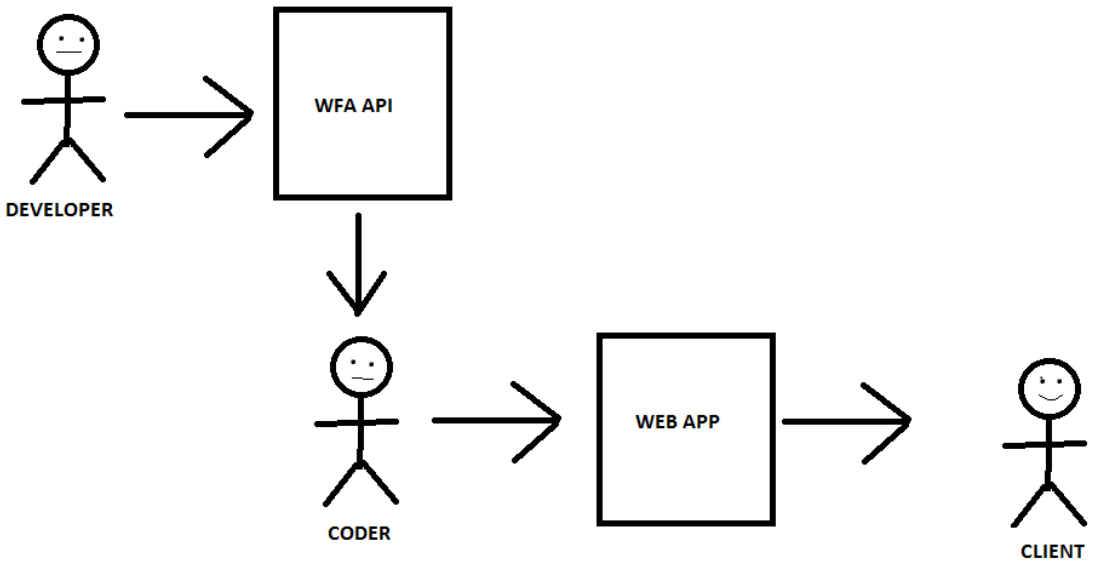


Figure 1. System Environment

2.2 Specifications

The following section outlines the modules and use cases of the API which the Coder can use:

2.2.1 LOGIN MODULE:

a) Setup Login Page:

Required to set up a basic login page consisting of the fields: webmail and password, in one PHP command. All necessary validation on passwords and the webmail field is done by the module only, requiring next to no input from the coder.

b) Connect External Credential Database:

Required to connect an external database of credentials (usually webmail and passwords) to the WFA framework form. Authentication can be done in a secure way and databases encrypted using popular hashing conventions such as MD5 are supported in addition to the plain-text databases.

c) Login/Logout Users:

Required to validate user input and check their credentials against the credential database. It uses session variables with information about the user which are then passed on to the main request form. This also decides where in the hierarchy the current user lies.

2.2.2 FORM BUILDER MODULE:

a) Build Request Form:

Required to set up the initial properties of the form such as the method (GET or POST) and custom CSS if wanted. The sub-utilities (and somewhat principle ones too) include:

> **i) Add Form Element:** Required to add input elements to the form. The coder has to supply the type of input element and the name under which this data should be saved. In case of misuse of this utility, there is a provision for error caching.
>
> **ii) Add Form Rule:** Required to add validation rules to the input elements. Instead of writing lengthy JavaScript or PHP validation scripts, the coder can just mention that a field requires a particular format of validation. Provided formats include required, email and strictly numeral field validation rules.

b) Submit Request:

Required for the submission and validation of form data. All the rules applied are cross-referenced, and all the errors are cached and shown to the user. After all validation checks have been successfully completed, the form data is then sent to the database.

c) Create Request Management Database:

Required to create the database for request management. The coder has to specify the database fields and their appropriate names and names of associated input elements and they will be linked automatically.

2.2.3 HIERARCHY MODULE:

a) Define Hierarchy:

Required to define a particular hierarchy structure required for request management. The coder can define as many levels as needed and the requests will need to be sequentially approved by all the people in the hierarchy.

2.2.4 REQUEST HANDLING MODULE:

a) Update Database After Change in Request Status:

Required to update the database with the current status of the request and up to where have the requests progressed in the hierarchy.

b) Notify Client:

Required to notify the client of the status of request via mail or notifications on a dashboard (if the coder wishes to implement this).

3.0 Non-Functional Requirements

The WebApp, built on top of the API will require a server with high speed internet capability. The physical machine required to host the WebApp will be determined by coder. The software developed here assumes the use of Linux machine installed with PHP, MySQL for the purpose of hosting the WebApp. The speed of Client's connection will depend on the hardware used rather than characteristics of this system.

3.1 Security Requirements:

The server on which the WebApp is hosted will have its own security to prevent unauthorized requests. There is a restriction on access to the Credential Database or Request Management Database. There is no special protection built into this system other than to avoid malicious requests. The use of email to notify the client about status of their request is external to the system.

3.2 API Testing Requirements:

The API Testing has to be performed for the system. During Testing, a verification of following things has to be looked at.

- Return Value based on input condition - The return value from the API's are checked based on the input condition.
- Verify if the API's does not return anything.
- Verify if the API triggers some other event or calls another API. The Events output should be tracked and verified.
- Verify if the API is updating any data structure.

The API testing for the software will be done with help of PHPUnit.

3.3 Database Requirements:

The software assumes the presence of the Credential Database with the Coder, and has to be connected using appropriate functions from Login Module of the API. Credential Database has to be provided by the organization/institute. Hence, no provisions are made for editing the Credential Database.

The Request Management Database is created and maintained by the software itself depending on the specifications given by the Coder while using the Form Builder Module of the API.

3.4 Documentation Requirements:

The software requires a proper and extensive documentation, so that Coder can easily understand and use the features made available to him by the API.

For the very purpose of Documentation, phpDocumentor is used. phpDocumentor is a tool that makes it possible to generate documentation directly from your PHP source code. It is required to follow the standard conventions for phpDocumentor while writing the code.

3.5 Maintainability and Reliability Requirements:

Any change in the codebase of the API has to satisfy minimum standards of following requirements which will aid maintainability and reliability.

- Naming Conventions - PSR standard
- Documentation Conventions - phpDocumentor standard
- Tab stops – four.

4.0 User Characteristics

The coder is expected to have a very basic knowledge of PHP. The API has coder-friendly and easy to understand and use functions. Anyone with minimum programming experience can use the API to develop WebApps based on Workflow Automation. All the features available for use are well documented in coder manual for API.

The Client is expected to be Internet and Windows literate. He should be able to use button, pull-down menus, and similar tools.