Chapter 3

Software Requirements Specification

Purpose

The principle reason for setting up this report is to give a general knowledge into the examination and necessities of the current framework or circumstance and for deciding the working qualities of the framework.

Scope

This Document assumes an elementary job in the improvement life cycle (SDLC) and it portrays the total prerequisite of the framework. It is created for use by the designers. Any progressions made to the prerequisites later on should experience formal change in the agreement process.

SDLC

For the most efficient and iterative working principles, the Agile Methodology had been chosen for the complete development purpose. The complete development process was divided into four sprints, and a scrum backlog was prepared. After each sprint, the sprint backlog was being compared with overall development.

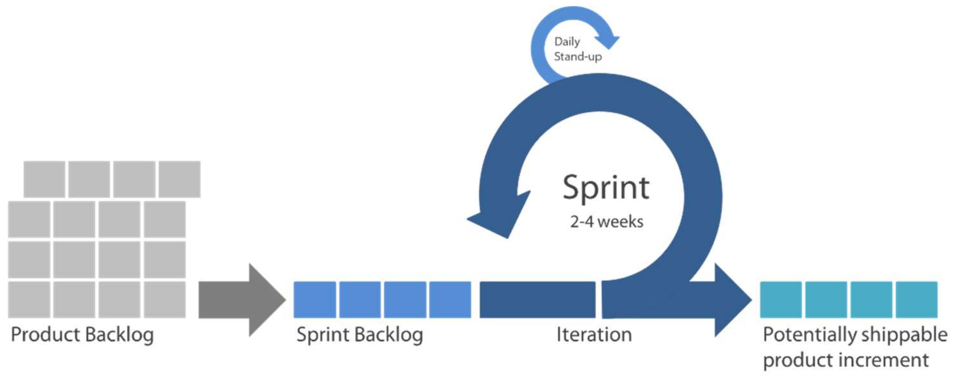

Fig 1. Scrum Flow

Timelines (Scrum Backlog)

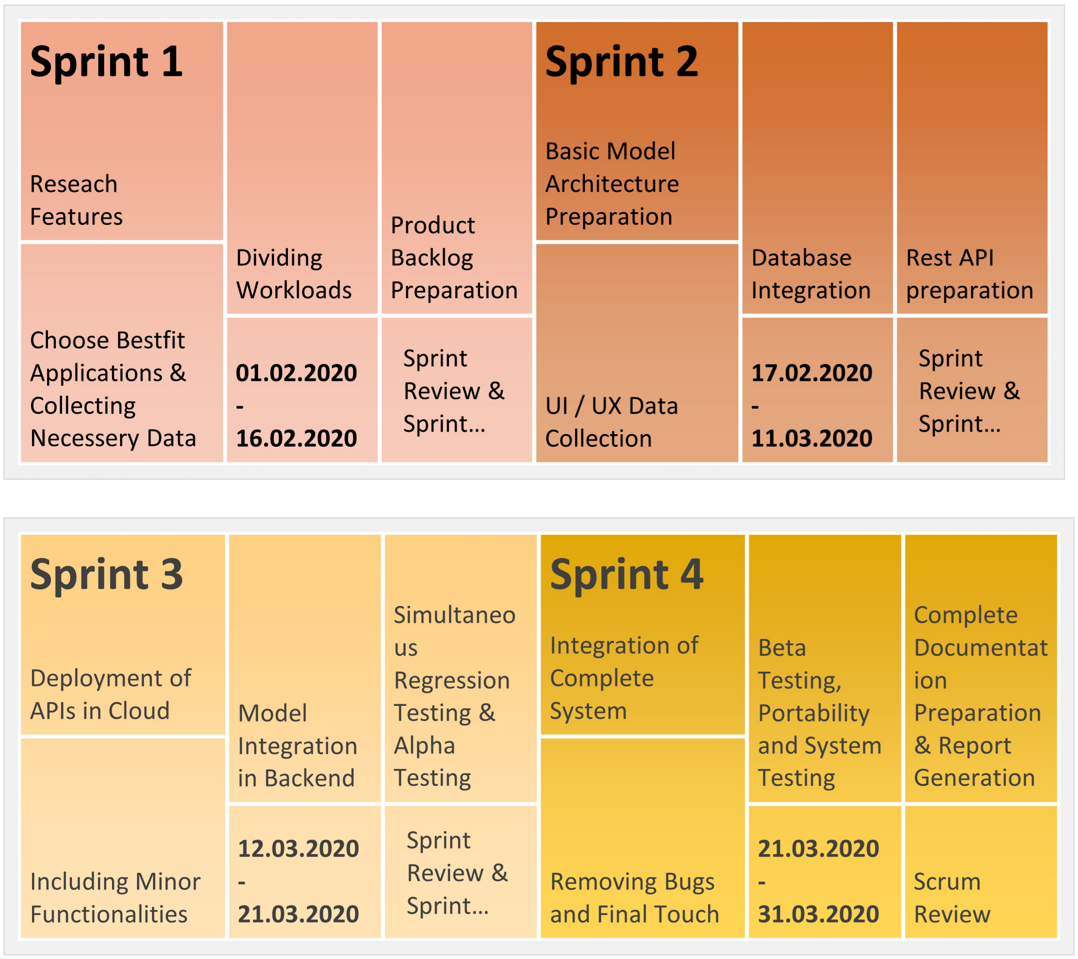
Sprint 1
Reseach Features
Choose Bestfit Applications & Collecting Necessery Data
Dividing Workloads
01.02.2020 - 16.02.2020
Product Backlog Preparation
Sprint Review & Sprint...
Sprint 2
Basic Model Architecture Preparation
UI / UX Data Collection
Database Integration
17.02.2020 - 11.03.2020
Rest API preparation
Sprint Review & Sprint...
Sprint 3
Deployment of APIs in Cloud
Including Minor Functionalities
Model Integration in Backend
12.03.2020 - 21.03.2020
Simultaneous Regression Testing & Alpha Testing
Sprint Review & Sprint...
Sprint 4
Integration of Complete System
Removing Bugs and Final Touch
Beta Testing, Portability and System Testing
21.03.2020 - 31.03.2020
Complete Documentation Preparation & Report Generation
Scrum Review

Fig 2. Sprint Burn down Chart

3.1. Developers' Responsibility Overview

The developers are responsible for:

1. Developing the framework, which meets the Software Requirements and understanding every one of the necessities of the framework.
   a. Taking care of simple but attractive user experience.
   b. Considering the modularity of each frame for avoiding ambiguity during integration.
   c. Performing alpha test in every stage of development.

   d. Demonstrating the framework and introducing the structure at customers' areas after the acceptance testing is effective.
2. Integrating encapsulated modules which are inaccessible to user.
   a. Performing regression test while integrating.
   b. Demonstration of partially developed system to perform beta test.
3. Using the new data to reform backbones.
   a. Keeping models up-to-date.
4. Submitting the required client manual depicting the framework interfaces to deal with it and furthermore, the archives of the framework.
   a. Managing any client preparing that may be required for utilizing the framework.
   b. Keeping up the framework for a time of one year after installation.

3.2. Environment

For the complete development, systems of two different configurations were used.

1. The analysis, classifier preparation has been implemented using Python 3.7.7, and GUI using PHP 7.4 on an Intel(R) Core(TM) i5 7200U CPU 2.50GHz with 8GB RAM and 64 bits Windows 10 Home operating system.
2. The deployment has been implemented using Linux kernel and Docker on an Intel(R) Core(TM) Xeon CPU 1GHz with 1GB RAM and 64 bits Ubuntu 16.04 operating system.

3.3. Tools Used

- Semantic Markup
- PHP
- CSS
- JavaScript
- Owl Carousal
- jQuery
- Bootstrap 4
- Geocoding
- Python 3.6.9
- Flask
- MongoDB
- Mongo Atlas
- Ajax
- XAMPP
- XML
- Java SE
- Flask & Dart
- Pickle Tool
- Defined Classifiers
- Tensorflow v1.8
- Event Control
- Heroku Toolbelt
- Heroku Cloud
- AWS IAM
- AWS EC2
- AWS S3 Bucket
- Canvas.JS
- Postman
- PuTTY
- PuTTY Gen
- WinSCP
- Chrome DevTools
- IP Fingerprint
- Particles.JS
- Express
- Docker & Procfile
- Socket.IO
- Android Studio
- Google Colab
- VSCode
- Git Bash
- Anaconda
- AR.JS

3.4. System Hierarchy

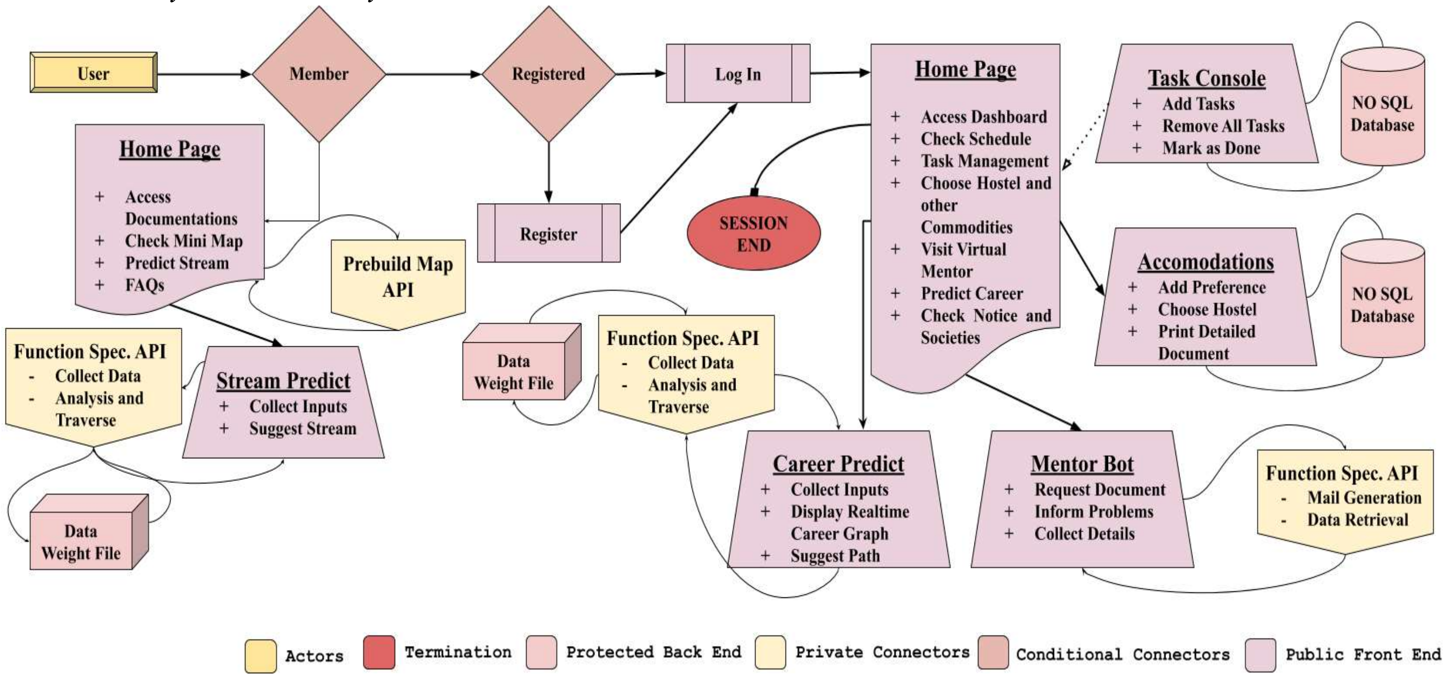

Fig 3. System Hierarchy