Team Hotel Scrum

Sheelendra Singh (Matriculation number:- 2412861)
Tushar Wankhede (Matriculation number:- 2354495)
Anagha Sagare (Matriculation number:- 2654182)
Ankush Chikhale (Matriculation number:- 2973449)
Dhanasekar Boopalan (Matriculation number:- 2863180)

TECHNISCHE
UNIVERSITÄT
DARMSTADT

1. Table of Contents

2. Introduction

SCRUM is defined as "a framework for developing and sustaining complex products". It is an iterative and incremental agile framework used for handling and completing complex projects. At first Scrum was framed for software development project, but it works well for any complex, innovative scope of work. Scrum originally was formalized for software development projects. The Scrum framework is inherently very simple. Scrum has been successfully used and transforms project management across every industry, business and even across life in general. Using scrum in project makes entire project more agile (able to move quickly and easily), discovering new ways to react quickly and respond more accurately to the inevitable change that occurs in project, and by staying focused, collaborating, and communicating, you can accomplish what truly needs to be done successfully.

This design document gives the narrative and graphical documentation of SCRUM. This document includes technical description and demonstrates the user documentation that briefly describes how to use the application.

2.1. Purpose

In any project, scrum brings flexibility, stability, cohesion, order and ultimately success. Organizations struggle to organize and compete in today’s economy, can be helped with building projects within time frame, scrum provides a civilized means of rapidly bringing new, complex, high-technology products to market.
Scrum enables people to work harmoniously for their mutual benefit while producing some of today’s most complex, sophisticated products. Scrum is the social engineering of today’s enterprise for the co-operative fulfillment of all involved.

2.2. Scope

This documentation gives detailed information about the used Technologies for building the website. It tries to explain detailed flow of project which zoom into technical details of each and every part of interaction from login to saving data in database. The documentation will be helpful for any technical person to overtake the project and further add on new functionality. The knowledge Transfer details are being mentioned to make the life of developer easy of understanding flow at each step and the technology used.

2.3. Definitions, Acronyms, and Abbreviations

- Scrum Master: In scrum, scrum Master is responsible for ensuring scrum is understood and enacted. Scrum master is responsible for service to product owner.
- Product Owner: In scrum, product owner is one who is responsible for maximizing the value of the product and the work of the development team. Product owner also takes care of Product Backlog Management
- Development Team: In Scrum, development Team takes care of all the development in Project and is self-organizing.
- Node is: Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. Node.js' package ecosystem, npm, is the largest ecosystem of open source libraries in the world.
- JavaScript: It is a high-level, dynamic, untyped, and interpreted programming language.
- Hypertext Markup Language, commonly referred to as HTML, is the standard markup language used to create web pages.
- AngularJS (commonly referred to as "Angular" or "Angular.js") is an open-source web application framework mainly maintained by Google and by a community of individuals most notably, Rangle.io as well as group developers and corporations to address many of the challenges encountered in developing single-page applications.
- Cascading Style Sheets (**CSS**) is a style sheet language used for describing the presentation of a document written in a markup language.
- Web storm : Lightweight yet powerful IDE, perfectly equipped for complex client-side development and server-side development with Node.js
- REST: It is the software architectural style of the World Wide Web. REST's coordinated set of constraints, applied to the design of components in a distributed hypermedia system, can lead to a higher-performing and more maintainable software architecture.
- Express Js: Express.js is a Node.js web application server framework, designed for building single-page, multi-page, and hybrid web applications. It is the de facto standard server framework for node.js.

3. Design Overview

3.1. Introduction

Scrum is one of the most used project management methodology in Industry today. Making a working website application of Scrum is of great help to the organization. The management of each and every small details with the help of this website will be at great ease and it will definitely make the life of project manager aka product owner and team member easy. In this small project we managed to create a web based application for scrum. The website undertakes all the basic requirement of scrum such as two main users as product owner and team member.

The website also allows product owner to add or delete project, team members and user stories. Once added team member has rights under added project to create task and assign task. The website has also added chat feature for each particular project. It makes life of particular project team easy to communicate and keep up to date by clicking on particular project. As single manager (Product Owner) can handle more than one project and same with team members. One member can be assigned to more than one project, bifurcation with keeping project at root node makes the classification easy and hence Scrum can be easily achieved for a particular project.

3.2. Components

Components involved are:-

- Angularjs
- Node JS
- MongoDB
- Mongoose
- Web storm

3.3. System Architecture

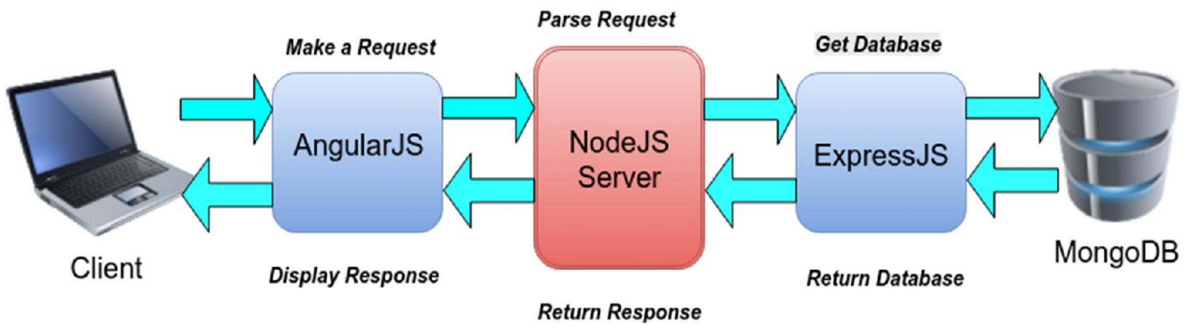


Fig: 1.System Architecture of Scrum Website

When client first interacts with website as login, the initial technology which uses in front end is Angular JS. Angular JS in turn makes interaction with NodeJs server to run Mongoose queries and routes so that different CSS pages are loaded. NodeJS interacts with ExpressJS for interaction with MongoDB. ExpressJS uses Mongoose running on NodeJs and establishes connection with MongoDB. The authentication and stable database MongoDB is used for saving all the relevant data for the project.

3.4. System Interface

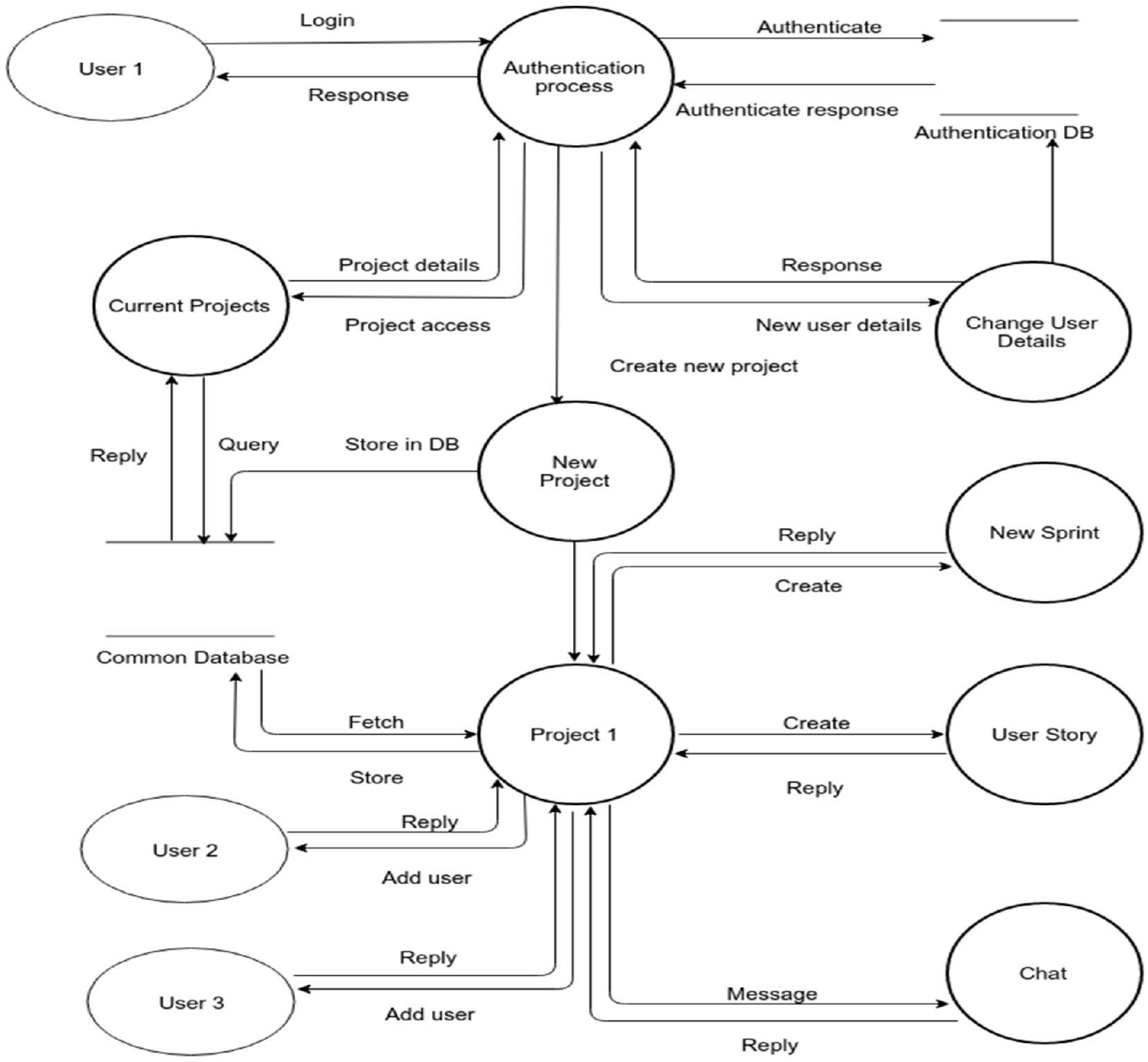


Fig: 2.System Interface Design of Scrum Website

3.5. Detailed Design

For the scrum web application, different users will be able to login. The roles assigned are Project Manager i.e. Product owner, Developer and Team member. Depending on the Role and the authentication, the home page will be showed up. If the user is product owner, he will have options to create a project. If there is team member login, only Projects to which he is assigned will be showed and Team member will not have any admin rights to create a project.

Given diagram above in the Interface section, any user irrespective of roles has to register/login and the login authentication will be authenticated by checking from database (MongoDB) and fulfilling the constraints. At the same time role mentioned for the user will be fetched from the DB and the relevant home page will be shown to user. Logged in user can change the details of his/her account by going to change user details. If the user is Product owner, he/she can create project and add members to the project, this can be done by going in “New Project”. Product owner can also check which all projects are under his/her belt. After product owner adding team members, respective team member once logged in can see the notification with the project they are added in. Product owner & team members has options to create sprint and add stories. This can be done under umbrella of project assigned to team member or project created by product owner.

Adding to project is the chat section added. The chat section is separated for each project. Every project has its own chat section and that is fetched through DB. For a particular project creating sprint has new level2 Data flow which is linked to particular project and it can also be accessed using the common Database.

For each module like login, register, project creation and modification, sprint, chat, product backlog we created separate html pages. All script, css references are loaded inside index html.
Root directory of the project looks like this:

```
app.js
bin/
node_modules/
package.json
public/
routes/
views/
```

Brief information:

- app.js - This is starting point of the app. All the modules, routes, database connections are imported here.
- bin/ - This directory by default contains file www which includes app.js and starts node.js server when script inside it is invoked.
- node_modules - All the external modules installed using npm install are here.
- package.json - This file defines a JSON object that contains various properties of our project including things such as name and version number. It can also defines what

versions of Node are required and what modules our project depends on. A list of possible options can be found in npm's documentation.

- public/ - All files publically available to server are placed here like css, js, templates.
- routes/ - Used to store Node controllers and backend code e.g. we have outes/api.js, routes/sprintapi.js etc.
- views/ - we can keep our views here.

1. Front end part:

In the project we have used different angular directives we will see some of their usage.

1.1 ng-controller:

In project: We have created controller for each task e.g. for adding new project we have `app.controller('addProjectController', function($scope, $rootScope,$http, $location,$window ) {}` in registration.js. Where we add `ng-controller="addProjectController"` this in html.

1.2 Ng-model: binds the value of html control like input,textare to application data.
E.g `<input type="text" ng-model="project.projectName" class="form-control" id="usr">`

1.3 Ng-repeat: as suggests it repeats html element. E.g. repeats for all the projects in the entire list
`<li ng-repeat="projects in projectLists" >`

1.4 ng-show: shows or hides the html element.
E.g. Project list is shown to only authenticated user `<li class="dropdown" ng-show="authenticated">`

1.5 Ng-click: redirects to the code which will be executed when that it is clicked
`<a ng-click="signout()" >Signout</a></li>`

Angular services: As controller scope is ended data is no longer available and to used one controllers data in another we need angular services.
In project: We have many factories created in project. E.g. we have created factory `app.factory('projectReference', function () {}` in Registration.js. This `'projectReference'` is then injected in controllers so that we can access the data.

ng-Enter- We used this for submitting the data on enter keypress.
In project: We are using this for login and on chat html page for submitting login credentials and chat on enter. E.g.`app.directive('onEnter', function() {return function (scope, elm, attr) {elm.bind('keypress', function (e) {if (e.keyCode === 13) {scope.$apply(attr.onEnter);}});};});`
We used this in Registration js and in html we add `on-enter="function to call"`

Using external modules: As we are using some external modules in the project we use `var app = angular.module('registration', ['ui.router']);` and added library source to index.html
`<script src="http://angular-ui.github.io/ui-router/release/angular-ui-router.min.js"></script>`

In order to configure this in angular js we use .config() function to set up state. e.g.*app*.config(function($stateProvider,$urlRouterProvider,$httpProvider,ChartJsProvider)

2. Back end part:

Creating schemas with mongoose:
We added following code in app.js to connect to local instance of MongoDB.

```
var mongoose = require('mongoose');
mongoose.connect("mongodb://localhost:27017/MyDB");
```

In our /models directory we created model.js. All the required schemas are written here.

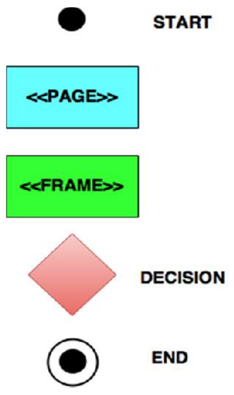
3. Routes:

Here some routes are created which frontend client can interact. We have routes/api.js, routes/sprintapi.js etc.

4. Passport:

Uses different authentication methods that are provided as a separate module. We used passport-init here. Added `var passport= require('passport');` this in app.js
Created angular service for it.

4. User Interface Design



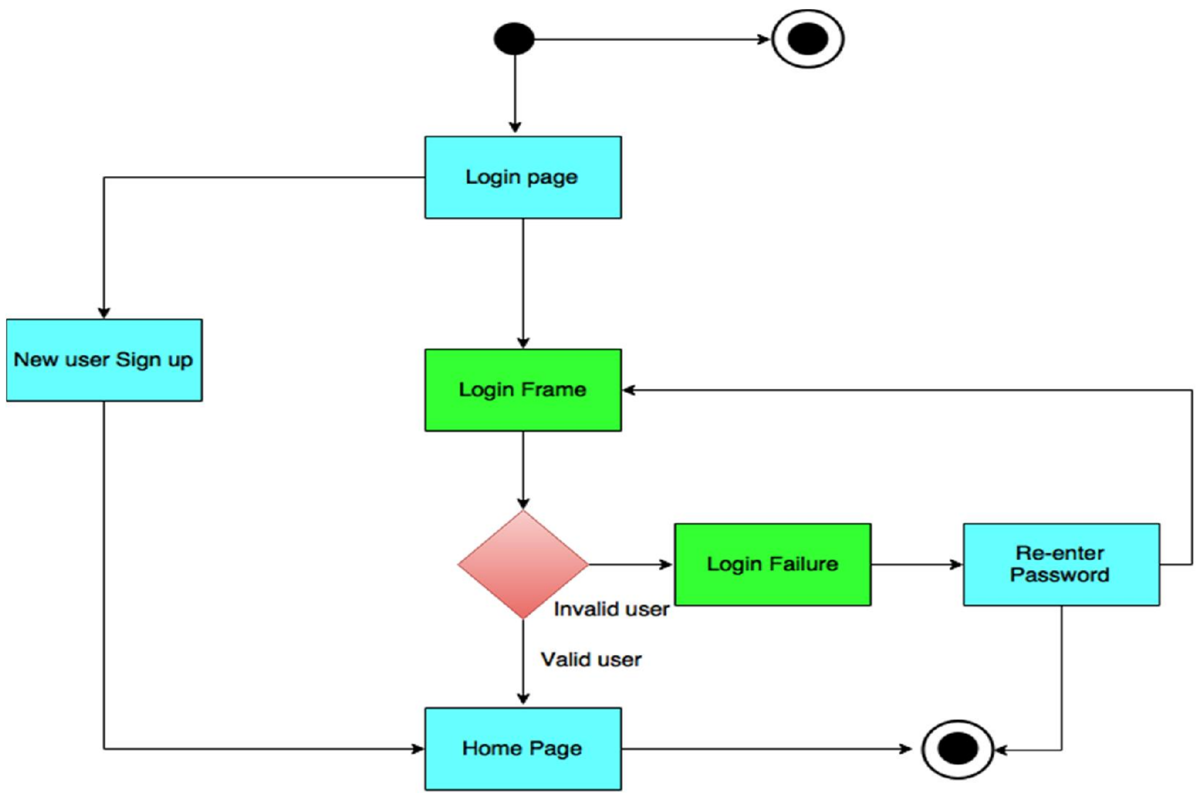


Fig: 3.User interface for login page of scrum

Whenever user clicks the url of a website the login page is first viewed. Now from the login page we have two navigation pages, user can either login or Signup. Clicking on Login, a Login Frame will be viewed and if he/she has a valid user id and password and redirected to Home page or the same frame of Login with error message appears and user needs to re-enter the credentials. User can also click on signup button and give details and then redirected to Home page.

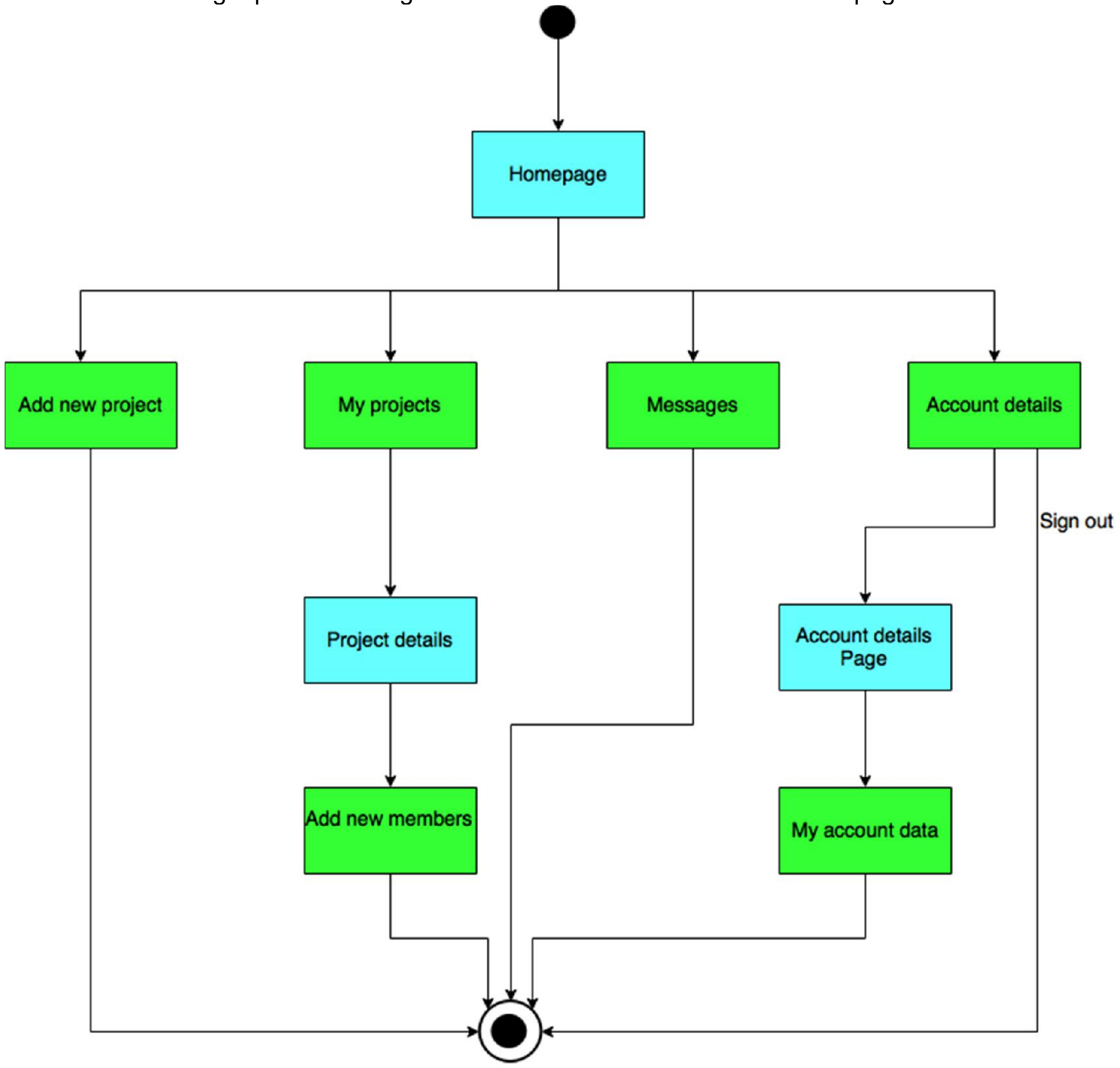


Fig: 4. User interface for Home page of scrum

In the Home page there are four frame available to navigate as add new project, My Projects, Messages, Account details. Clicking on My project will redirect to Project detail page and also add new Members page. Account Details contains Account details page and my account data frame.

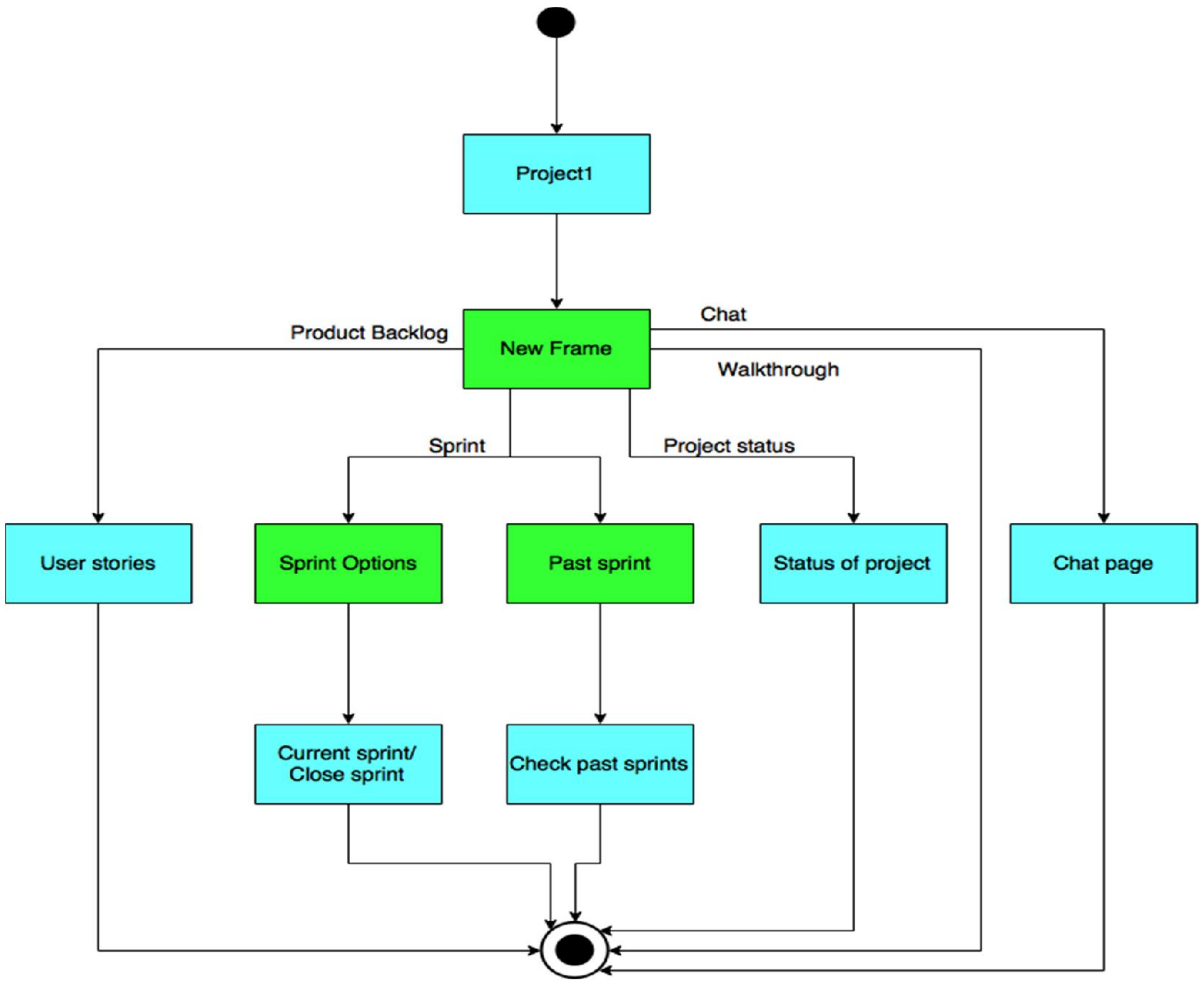


Fig: 5. User interface after selecting new/existing project in scrum

Once user clicks on Add new project, new project with a project name will appear. Here for the sake of explanation Project 1 is taken into consideration. Once user is on Project1, New frame with different options are available. These new options will lead to new Frames and new pages. Various frame and pages under Project1 available are “User stories”, “Sprint Options”, “Past sprint”, “Status of project”, “Chat page”.

5. Conclusion:

The newly developed SCRUM application has achieved all of the goals mentioned in the
Project requirement documentation. It enables users to manage project and apply SCRUM functionality into project. It also enables product owner to add, delete and modify project. Moreover, within a project, different users can be added. We have also provided Scrum documentation for future modification and user interface for new user.

6. References:

1.https://www.mongodb.org/
2.https://docs.mongodb.org/manual/?_ga=1.126879822.214804439.1457126589
3.https://angularjs.org/#
4.https://en.wikipedia.org/wiki/JavaScript
5.http://www.w3schools.com/js/
6.https://www.mountaingoatsoftware.com/agile/scrum
7.https://en.wikipedia.org/wiki/Scrum
8.https://nodejs.org/en/
9.https://www.jetbrains.com/webstorm/
10.http://mongoosejs.com/
11.https://en.wikipedia.org/wiki/Representational_state_transfer
12.http://expressjs.com/