**SECOND INCREMENT REPORT**

PROJECT NAME: GO EASY

PAVAN KUMAR BOLLARAM (CLASS ID: 07)

PREETHAM KUMAR DANABOINA (CLASS ID: 11)

LAKSHMI PRIYANKA (CLASS ID: 49)

VARAPRASAD REDDY JAGGU (CLASS ID: 22)

**IMPORT EXISTING SERVICES/API:**

The following services are likely to be consumed in our project:

**Existing services/API**:

* http://api.eventful.com/rest/events/search?[...](http://api.eventful.com/docs/auth)&keywords=books&location=San+Diego&date=Future

The above API acknowledges a user about the live events happening across a specified location where the location can be retrieved from location and geo sensor of android mobile device

* <https://data.cityofchicago.org/resource/alternative-fuel-locations.json?>

This API also used to get the details of the location or city

* <https://maps.googleapis.com/maps/api/js?v=3.exp&sensor=true>
* This API is used for location tagging and navigation purpose which can be used by user in our application.

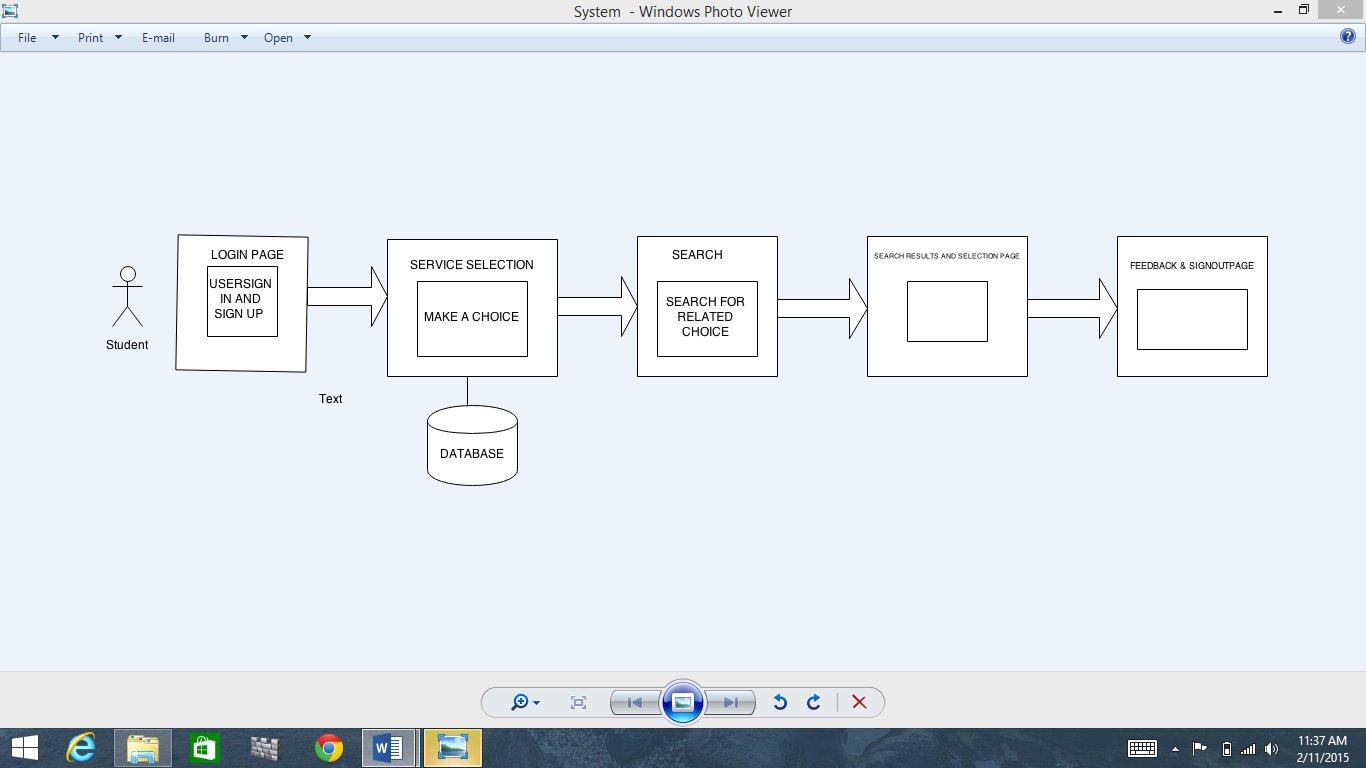
**Registration API**: developed custom registration Api which stores user login details into database table.

**Login API**: This custom api Deals with user login to application. It does validation with given user inputs

**DETAIL DESIGN OF SERVICES**:

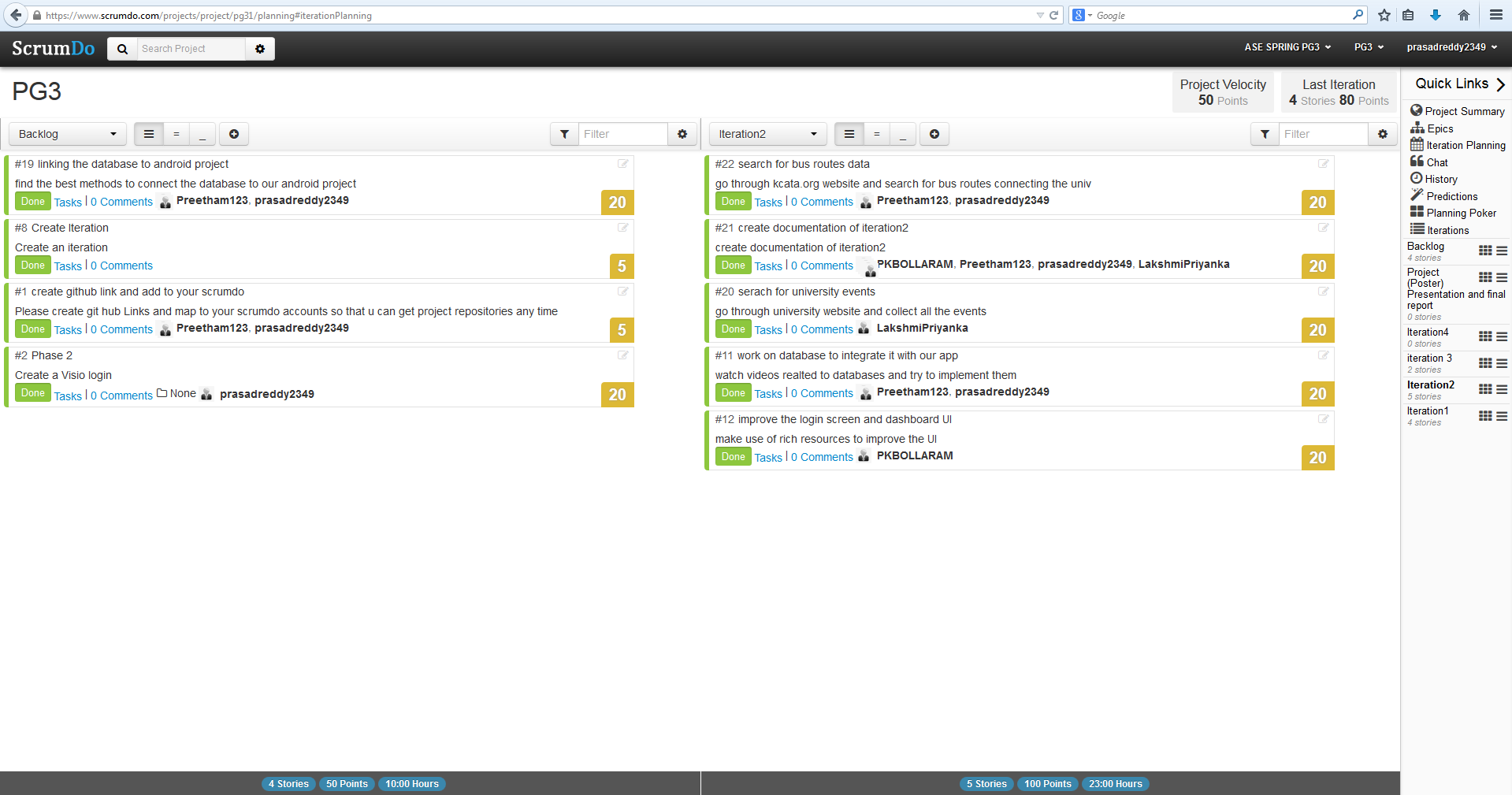
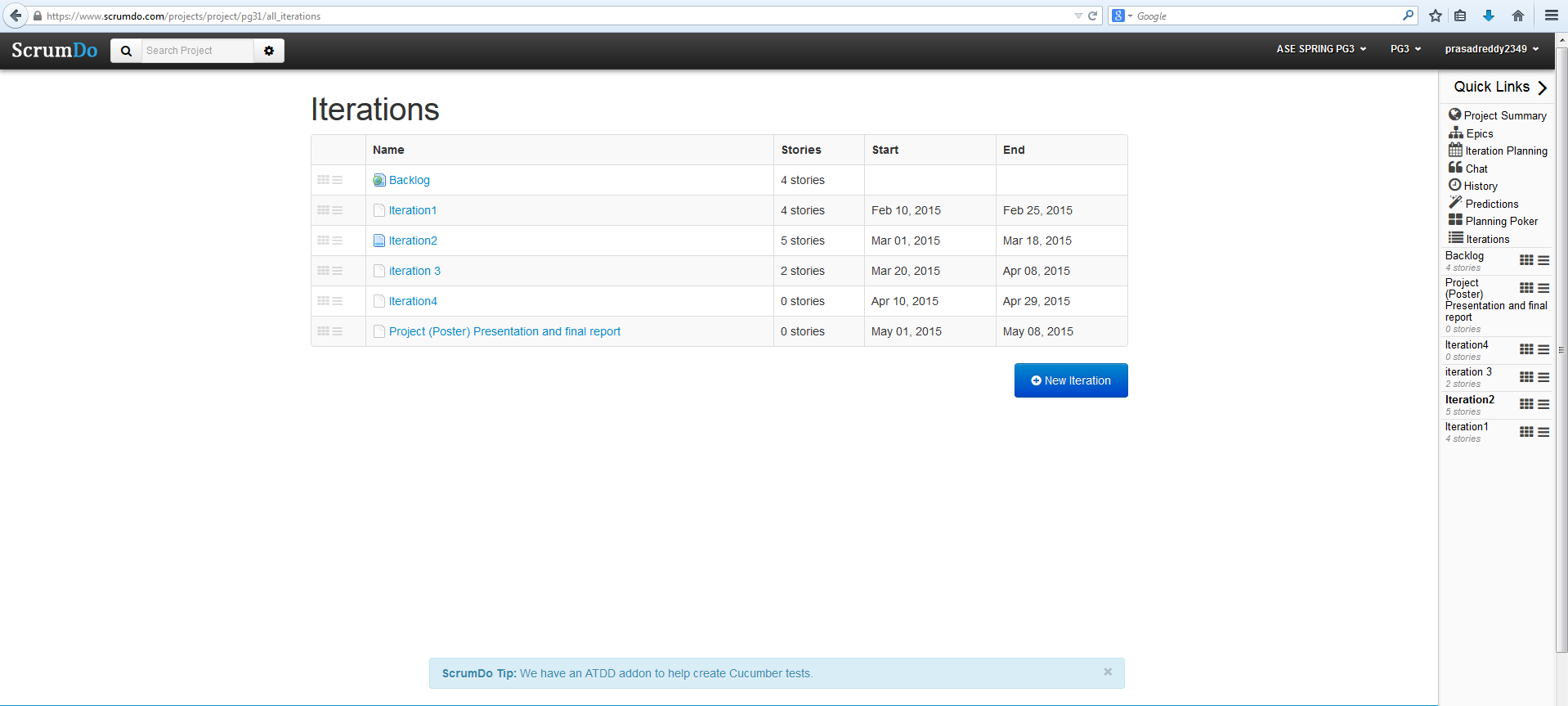
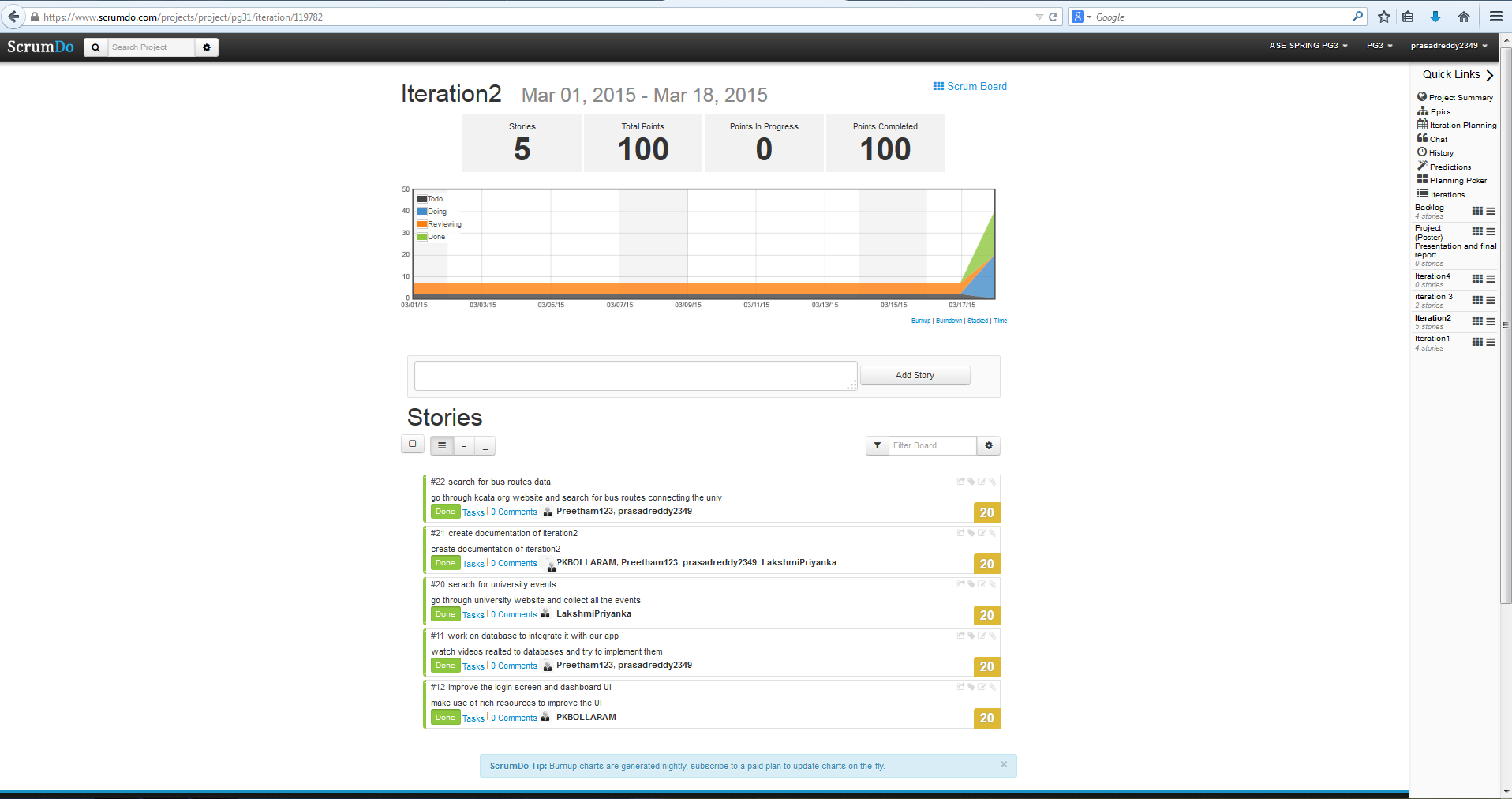
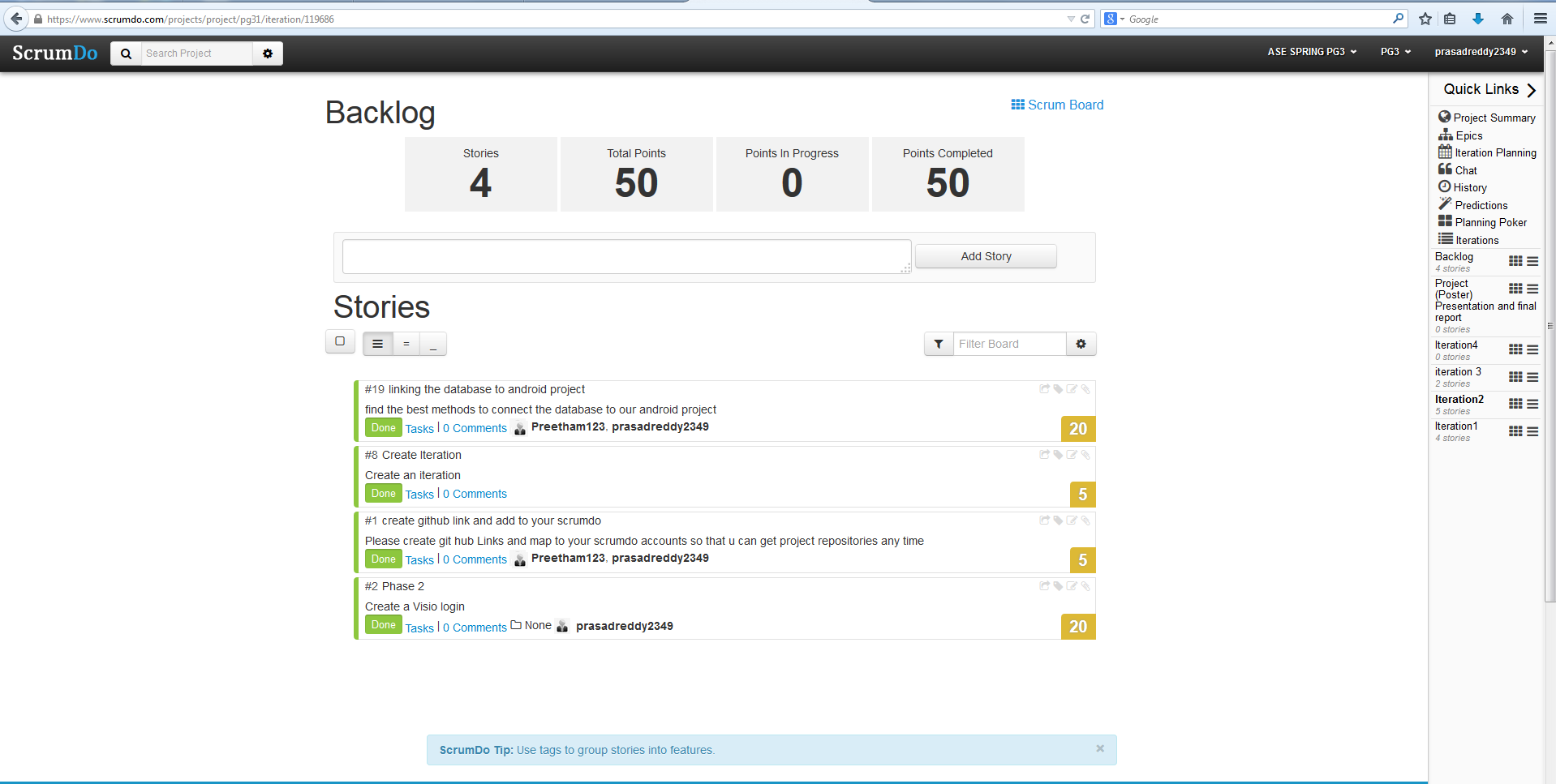
Go Easy Web services:

We are intended to develop customized web service which includes all the required data at one place which going to ease the way of searching for the events by end user. The below figure depicts the architecture which explains flow and behavioral description of each layer starting with login page of our application to the data layer.



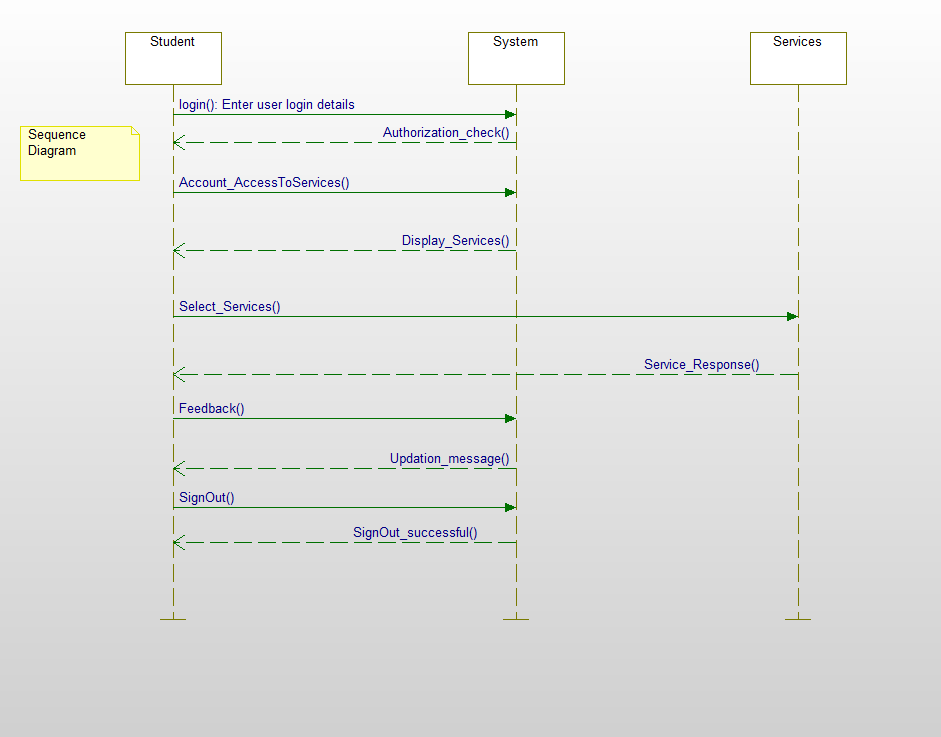
The application mainly consists of three section which includes entertainment related events, university related events and transportation related events. We are using publically available API to connect to their end point and get the response (usually JSON).

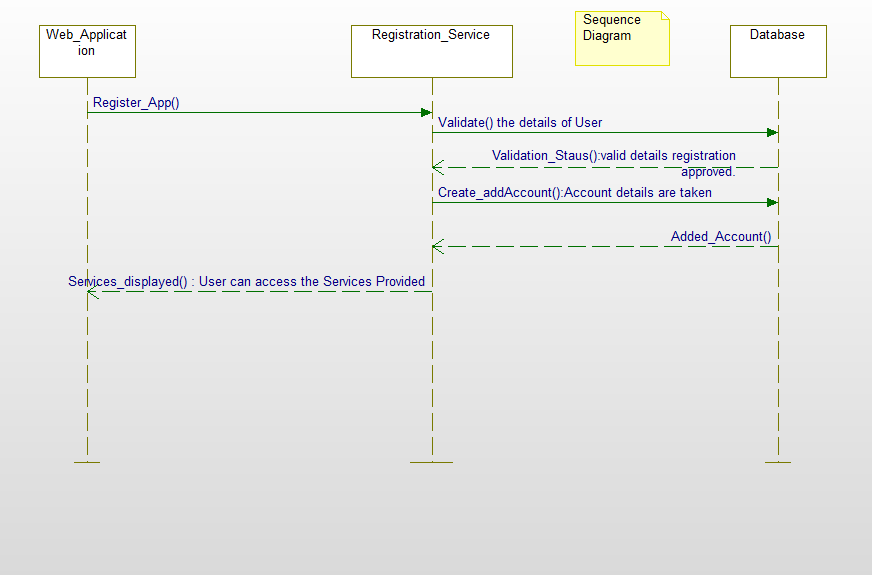
**Scrum Do and Stories:**



**IMPLEMENTATION:**

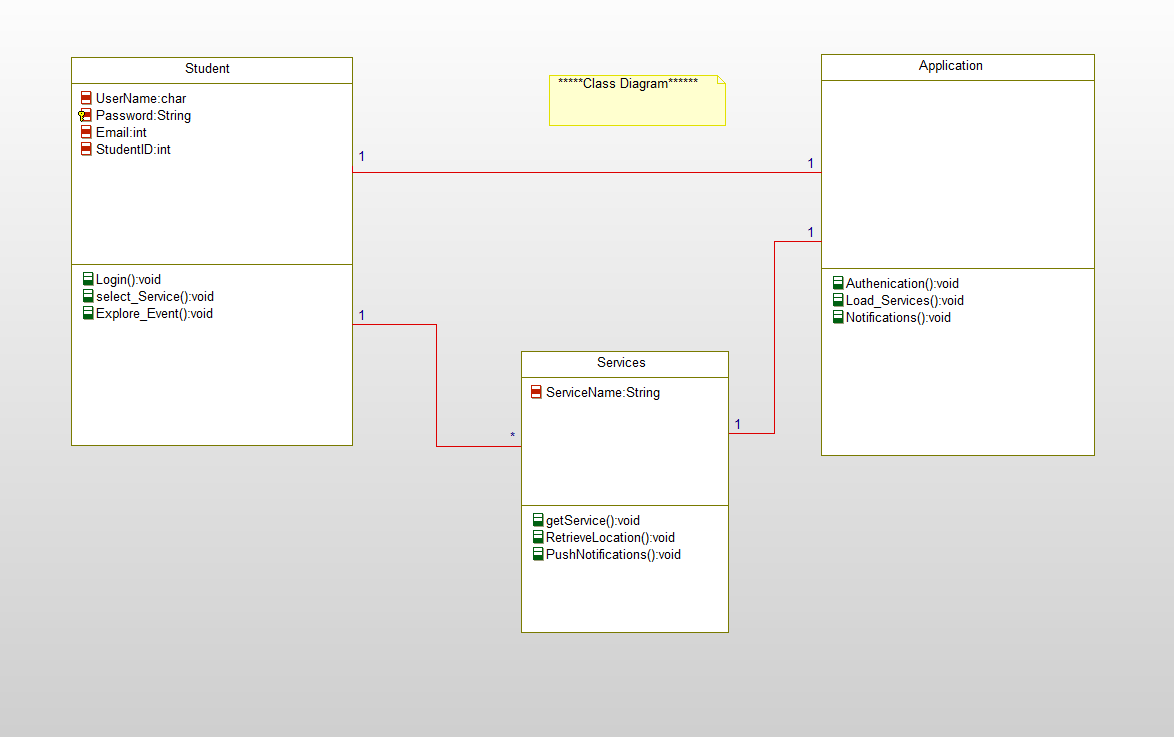
Sequence Diagrams:

****

****

Sequence diagram is message interaction chart diagram that depicts how processes order with one another and what process they do. It depicts the objects and classes interactions in time sequence. The above sequence diagram explains an abstraction level of system implementation with required services and flow of each service. User can register to events using Registration services offered by application which gets the users data and persists to database. This very same data can be fetched from the data layer using display services call and dashboard is updated with these results. Along with the useful live events data a small level navigation part is also going to be implemented in this application.

**Class Diagram:**



A class diagram is a type of structure diagram which describes the attributes, classes and methods (or operations) among the elements involved.

**TESTING:**

Tested web service availability through android

Tested user login validations with different probable conditions

Tested registration process with few specific conditions

**IMPLEMENTATION:**

Implementation of REST Services:

Two API’s are implemented as part of this increment. They are i) Login API

ii) Registration API

**Login API:**

Login API deals with login procedure which does validation on user inputs. Checks whether user is registered to application or not.

Separate database table is created which stores the registered details and does validations by fetching the details from this database table

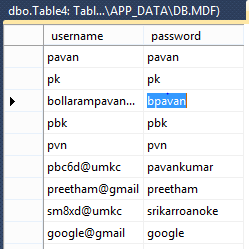


Table created to store Login details and Login API connects to this table and fetches data.

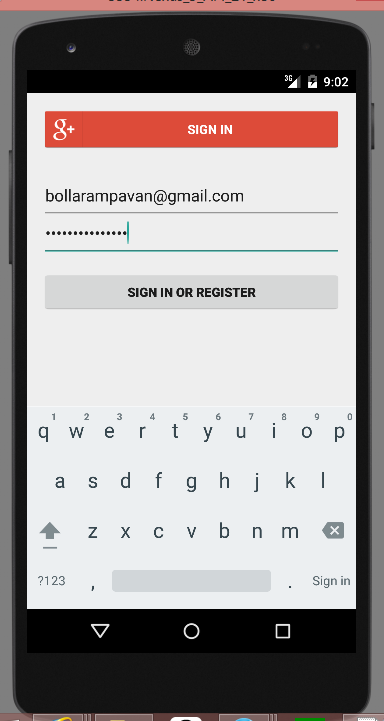
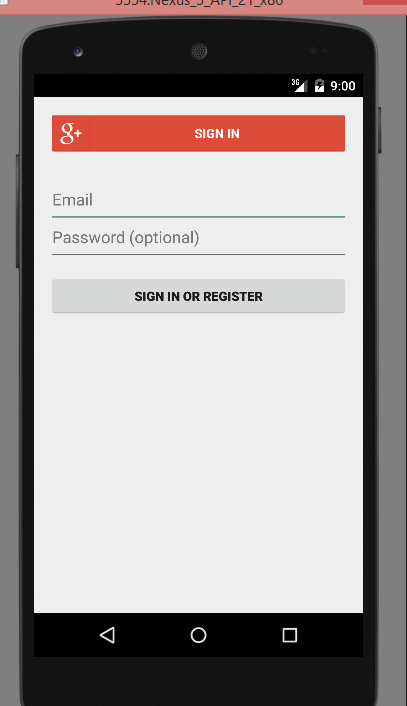
**Registration API:**

Registration API deals with registration of end user into application as of now we are accepting email id as primary attribute and password as second.

End user with email id and password which were given at the moment of registration can login into application.

**Implementation of User Interface (Mobile Interface):**

Screen I: Login Screen

****

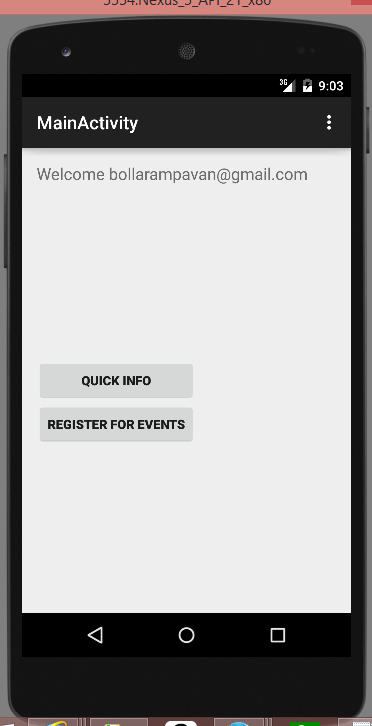
Login Screen

Login to system uses existing google+ sign in services so that end user can easily login with his/her Gmail id’s which reduces the use of separate database for our application .This login screen contains two separate parts which are

Sign in and Register: This button used incase user is already registered with our application. He can directly go into access of further screens. Or else if he is not registered already it generates into new screen showing you are not registered / wrong password; login again.

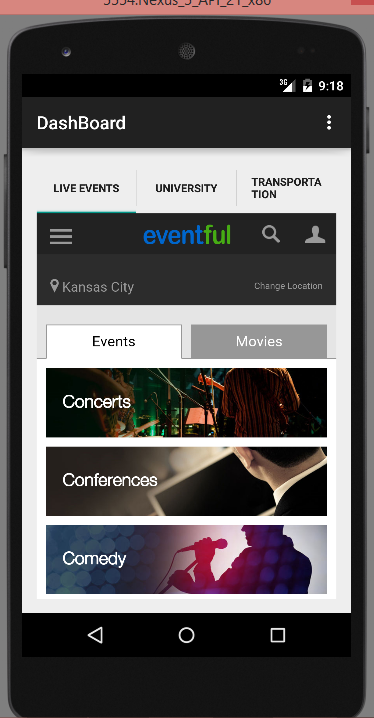
Email and Password: These fields which takes input from user and validates the user details which are already into the database when registered. Once validated user can login to application it pops into his dashboard.

**Welcome Page (Screen II)**

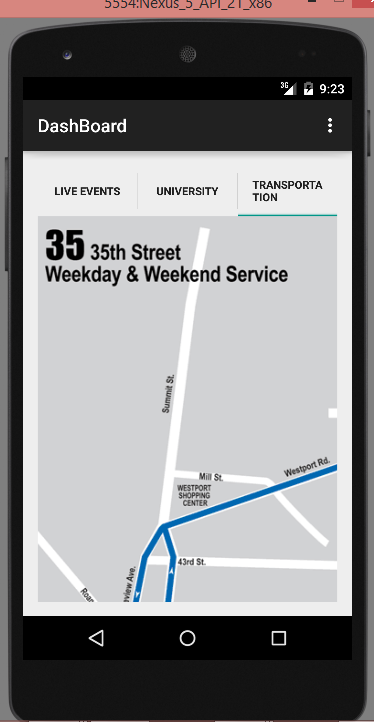


Welcome Page: The above figure depicts the welcome page displayed to the user when he / she login into the application. We included two Events buttons **Quick-Info** and **Register for Events**. On click of Quick Info, the button navigates into dashboard screen which is categorized into Live-Events, University and Transportation.

**Dashboard (Screen III)**



Live Event Tab.



Transportation Page

DASHBOARD

Dashboard: This is dashboard screen which will be displayed when user press the Quick info button in Screen II. It is completely mashup application where user can easily check the required tab.

We simplified GUI with simple tab host concept so that end user can easily navigate between Live-events, University and Transportation tabs. The content is displayed according to the selection of tabs.

**University (Screen IV)**

University details are presented to the user with tab in the Dashboard and when clicked navigated to different screen.

This screen consists of all data regarding university and it can be further improvised using deeper user interface

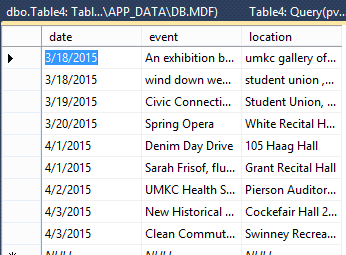
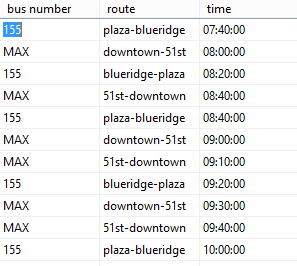
As of now we included few of the basic required data sections across screen they are as shown in the following images



The main dashboard screen consists of data sections like About Umkc, Map, Transportation, Academic details and sports sections.



**Database Design implementation**

Created database to store live events happening around umkc and web service is implemented to fetch these events

**DEPLOYMENT:**

ScrumDo link:

<https://www.scrumdo.com/projects/project/pg31/summary>

Git Hub Link:

**Source code-** https://github.com/pavankumar-b/ASEspringSem/Increment2

**Documentation-**[**https://github.com/pavankumar-b/ASEspringSem/IncrementDoc**](https://github.com/pavankumar-b/ASEspringSem/IncrementDoc)**2**

**PROJECT MANAGEMENT:**

Implementation Status Report

**Work Completed**:

**Description**

* Design of Android Login Screen.
* Successfully implemented Login Functionality using web API and database. Validation of login users with already registered email and password or even when registered with Gmail.
* We are Including Separate Simple tab host interface.
* Creating, validating and the database tables of user registered and while signing in.
* University, Transportation and Live Events tabs (On click) are implemented successfully.
* Tables for university live events and transportation tables were created.

**Responsibility**

Task 1: Implementation of web service / Login API

Pavankumar Bollaram/ Varaprasad jaggu

Task 2: Implementation of web service / Registration API

Preetham kumar / Lakshmi priyanka

Task 3: Database table creation

Pavan kumar Bollaram, varaprasad,preetham kumar, Priyanka

Task 4: Extended User interface design including dashboard screens

Pavan Kumar, Preetham kumar, Varaprasad jaggu, Priyanka

**Time Taken**

250 Man Hours

**Contribution**

Pavan Kumar Bollaram (25%)

Preetham Kumar Danaboina (25%)

Lakshmi Priyanka (25%)

Vara Prasad (25%)

**Work to be completed:**

**Task1:** User interface design for custom API events fetch

**Task2:** Improvise UI

**Task3:** Google maps integration and GPS location finding

**Risk Management:**

Very slow processing speed of android

Web services API needs to be in Run state always otherwise just throws 404 error.