**Project Description**

**Sparta-Scoop** is an attempt to ease the life of college students. It aims to address all those problems which are common for a college goer, and which can be solved through a network of students and alumni.

The idea is unique and has infinite potential because each year thousands of students join higher studies in the USA, and face common issues and concerns, such as deciding which courses to enroll for, which professors to choose, finding accommodations, jobs and internships. Such problems are amplified for international students, who are new to USA and are not familiar with the college life here.

Even though some websites have tried to address some of these issues, they haven’t been completely successful. On the other hand, SpartaScoop tackles all issues on a single platform, which is designed in a way that it can be extended and customized based on the needs of a specific college or university.

The developers of this solution are all international students, and have been through all the scenarios where community help and student cooperation are of ultimate help. Thus, we have aimed to incorporate all those features, which can be of help to a student.

The first version of our platform has been designed specifically for San Jose State University students, and we have limited ourselves to three important issues:

* Jobs and internships
* Accommodations
* On campus events

Based on the popularity of the initial release, we plan to incorporate several more features like:

* Course and professor reviews
* Buy and sell items
* Online tutoring

We also plan to release our platform for other universities, by customizing it to the needs and requirements of the specific university.

To prevent external influence, the website is only accessible to people who hold a valid email id of sjsu.edu domain.

**Requirements**

* Students with valid email address can sign up and use the platform.
* Registered users can make new posts and can view existing posts.
* All users can view the jobs posted on job page.
* All users can view the accommodations listed on accommodations page.
* All users can view the events listed on the events page.
* Every registered user can post accommodations, jobs and events.
* Posted accommodations are visible on google maps.
* Job posts need to have all the necessary fields as listed on the job posts.
* Events posts need to have all the necessary fields as listed on the portal.
* Admin can delete user profiles.
* Admin can view usage statistics, top 5 users and other parameters when logged in through admin-login.
* Admin can view the current website sessions and can monitor traffic, traffic locations.
* User can update his/her profile.
* Every user must have a unique identification email.
* User can search for a specific job, update, delete job posts.
* User can search for a specific accommodation, update, delete accommodation posts.
* User can search for a specific event, update, delete event posts.

**Web UI Requirement Principles**

**Voice of the Customer(VOC)**

Voice of the customer (VOC) is a term used in business and Information Technology (through ITIL, for example) to describe the in-depth process of capturing customer's expectations, preferences and aversions.

Specifically, the Voice of the Customer is a market research technique that produces a detailed set of customer wants and needs, organized into a hierarchical structure, and then prioritized in terms of relative importance and satisfaction with current alternatives.

Voice of the Customer studies typically consist of both qualitative and quantitative research steps. They are generally conducted at the start of any new product, process, or service design initiative to better understand the customer's wants and needs.

To capture the voice of our customers(students), we created a survey and took responses from students all over SJSU campus. The responses helped us evaluate the needs of SJSU students, and through the results of this survey, we could zero-in on our features for our first release.

Here are the results of our survey:

**Personas**

A persona is a fictional character that communicates the primary characteristics of a group of users, identified and selected as a key target through use of segmentation data, across the company in a usable and effective manner.

Sparta-Scoop is based on the following personas:

**Students:**

The primary user of the platform. This user uses the platform for gaining information posted on it by other students. The website must be able to provide relevant information every time a student logs in, he should also be able to share any information which can prove useful to other website users.

The student should not be presented with irrelevant information, neither should he be troubled with incorrect or fraudulent posts.

**Alumni:**

The students should be able to mark themselves as alumni or current students, based on whether they have graduated or not. Once graduated, the student should still be able to access the portal through an identification unique to the alumni.

**Job Shadowing:**

Job shadowing and contextual interviews are two techniques used to find out what people do, where they need help, and how your product can help them. To do that, we accompanied people while they do their jobs and talked to them about their jobs.

Following methods were used for job shadowing:

* Interaction with various students on campus.
* Interaction with various student communities.
* Discussing and analyzing problems faced by us as students at SJSU
* Creating a survey out of those responses.
* Discussing common problems and ideas with students and staff while on campus.
* Analyzing the results of our survey.

**Interfaces – RESTful & Server Side Design**

Representational state transfer (REST) or RESTful Web services are one way of providing interoperability between computer systems on the Internet. REST-compliant Web services allow requesting systems to access and manipulate textual representations of Web resources using a uniform and predefined set of stateless operations.

Sparta-Scoop has been developed in a completely RESTful way to provide extensibility and reusability, so that it can be modified for other universities as well.

The backend has been developed through **Node.js and Express.js.** The file server.js contains all endpoints which are provided by the server.

Following are the various endpoints which are used for interacting with the backend and share data:

* For getting all users:

app.get('/v1/users')

* Get user by email id:

app.get('/v1/users/:user\_emailid')

* create new user

app.post('/v1/users')

* delete user profile by email id

app.delete('/v1/users/:user\_emailid'

* update user profile - with email id

app.put('/v1/users/:user\_emailid')

* user login

app.post('/v1/users/:user\_emailid/login')

* validate unique username

app.get('/v1/users/:user\_emailid/checkunique)

* get my jobs

app.get('/v1/users/:userid/jobs')

* get my events

app.get('/v1/users/:userid/events')

* get my accomodations

app.get('/v1/users/:userid/accomodations')

* get all jobs available

app.get('/v1/jobs')

* get a particular job

app.get('/v1/jobs/:jobid')

* save a new job

app.post('/v1/jobs')

* delete a job

app.delete('/v1/jobs/:jobid')

* update a job

app.put('/v1/jobs/:jobid')

* get all events

app.get('/v1/events')

* get a particular event

app.get('/v1/events/:eventId')

* create new event

app.post('/v1/events')

* update event

app.put('/v1/events/:eventid')

* delete event

app.delete('/v1/events/:eventid')

* get all accomodations

app.get('/v1/accomodations')

* get particular accommodation

app.get('/v1/accomodations/:acmdid')

* post new accommodation

app.post('/v1/accomodations')

* put accommodation

app.put('/v1/accomodations/:acmdid')

* delete accommodation

app.delete('/v1/accomodations/:acmdid')

* get no of posts for each category

app.get('/v1/analytics/postcounts')

* get no of users from each branch

app.get('/v1/analytics/majorcounts')

* get no of accomodations according to price range

app.get('/v1/analytics/accomodationprices')

* get no of jobs according to pay

app.get('/v1/analytics/jobrates’)

* get no of events according to fees

app.get('/v1/analytics/eventscount')

**Design Patterns Used**

Following are the design patterns used in the project:

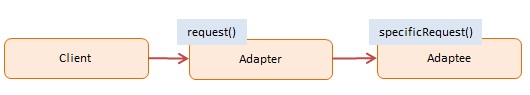
**Adapter:**

The Adapter pattern translates one interface (an object's properties and methods) to another. Adapters allows programming components to work together that otherwise wouldn't because of mismatched interfaces. The Adapter pattern is also referred to as the Wrapper Pattern.

One scenario where Adapters are commonly used is when new components need to be integrated and work together with existing components in the application.

Another scenario is refactoring in which parts of the program are rewritten with an improved interface, but the old code still expects the original interface.

The portal uses the adapter pattern when it saves and retrieves images from the database in the following fashion:



Uploads image Converts image to binary Receives binary

**Search Engine Optimization Provisioning**

Following methods have been incorporated in Sparta-Scoop for improving search engine optimization:

1. Provide a unique title for each page with important keywords near the beginning of the title.
2. Add a description meta tag to each page with a unique description for each page.
3. Name all the images with a related keyword. Use alt tags. Make images as small as possible. Use image optimizer.
4. Use .html links more preferably.
5. For all the image links, put keywords in the alt attribute text.
6. Keep the URL short and human readable. Avoid dynamic parameters whenever/wherever possible.
7. Use robots.txt to block the search engine from indexing few pages
8. Use rel=” next” and rel=” prev” in the <head> tag to indicate the relationship between the URLs
9. Declare language attribute in HTML page
10. Use different URL for desktop and mobile devices. rel=” canonical” and rel=” alternate”.
11. Remove all the scripts from the html pages.
12. Add social meta tag to the web pages
13. Increase page loading speed
    1. Minify code by removing unnecessary white spaces
    2. Use CSS sprites to reduce server requests
    3. Enable asynchronous downloading of external js files