Full Stack Web Application Project Proposal Template

**Team Name: Team Solo A**

**Application Name: GWlist**

**Coordinator:** Kaidi He G31860481

**Members:** *Jintong Jiang G48458308*

*Manuel Perez G36189218*

*Michael Rothkopf G21627706*

*Tingting Chang G32711691*

# Application Proposal Summary

The project will be a website where users can create profiles and sell items (usually second handed) to other users. The users can post images, description of the item they’re selling, price, their location, number of stock, and any delivery related information. When someone "buy" the item, the basic information of the seller is sent: email and phone number. The post of the item who was bought is hidden and no other user can buy it (if it was the last in stock). So, out of the application the users need to talk with each to finish the deal. After the item is "bought", users can grade each other with a positive, negative or neutral grade about the selling process.

# Team Management Strategy

*Briefly explain how you will be managing work loads among your team members. WIll you have a manager who delegates work? Will you have regular meetings? If so give me details. Let me know what you'll be doing to make sure the all the work is being done as expected.*

We divide the whole project into 3 parts: frontend, backend and database related. Both frontend and backend have 2 members to take charge. And database related things are owned by 1 of us who has experience before with MongoDB. We plan to meet once a week (not include the Thursday afternoon class) starting from next week. The coordinator is responsible to make the collaboration works and deal with the code base which is hosted on GitHub.

# List of Functionalities to be Implemented

*For each of the functionalities to be implemented, please provide detailed description of what the functionality will do and who's going to work on it and how long you think it will take.*

*Divide the functionalities so that it can be assigned to a single individual and the time can be estimated. It's OK if you miss a functionality. It's OK if you have to remove a functionality later. It's OK if you don't know the details of the functionalities. It's OK if you get the estimate time wrong. It's OK if the assigned owner changes.*

*Example Functionalities:*

Index Page – Front End

Using grids layout to display the basic item information including image (optional), name, brief description (optional) and a link to the item profile page. Also the user it belongs to will be added in the grid. Make columns unequal and use bigger part to display the item information. In the smaller part lists the top 5 rated users. Including their names and links to their profile page. Implementing a navigation bar on the top of the page which contains the button to login or registration and a search bar (reserved for search feature).

Owner: Kaidi He, Jintong Jiang

Estimated time: 1 week

Login Page – Front End

Basic HTML form let users to input the username (which is an edu email address) and password. And some JavaScript code to make the authentication with Web API which leads user to their profile page.

Owner: Jintong Jiang

Estimated Time: 3 days

## Create a User – Front End

The page contains a HTML Form that allows users to create a profile: a text field for a first name and last name, phone number and email address. Also some JavaScript codes that create a JSON object that could interact with the profile creation API.

Owner: Jintong Jiang

Estimated Time: 3 days

Create a User – API Back End

Web API that processes a JSON object representing a new user in the system. A user name will have a first and last name, a user ID, phone number, email address, and a rating associated with them based on their average rating of sold products. The Web API will pass this data to the DAO which will persist it into a user database table

Owner: Michael Rothkopf

Estimated Time: 1 week

Update User Profile – API Back End

Web API that updates a user profile in the system. The Web API will process a JSON object containing the updated values for a user. This will be passed to the DAO which will update the user table in the database.

Owner: Michael Rothkopf

Estimated Time: 2 days

## Create an Item – Front End

The page contains a HTML Form that allows users to create an item: text fields for item’s name, price, number of stock, location and delivery method. And there is an optional text field for user to make descriptions of their items. Also some JavaScript codes that create a JSON object that could interact with the item addition API.

Owner: Jintong Jiang

Estimated Time: 3 days

Add new Item to Sell – API Back End

Web API that allows a user to add an item to sell. The Web API will process a JSON object containing the details of the item including the name of the item, price, number being sold, location, and delivery information. The Web API will pass the data to the DAO which will update an Items table in the database.

Owner: Michael Rothkopf

Estimated Time: 3 days

Edit an Item a user is selling – API Back End

Web API that updates an item a user is selling in the database. The Web API will process a JSON object containing the updated values of an item. The data will then be passed to the DAO which will update the Items table in the database.

Owner: Michael Rothkopf

Estimated Time: 2 days

## Item Profile – Front End

Display all the information about an item. Plus, implement a “purchase” button in this page. The JavaScript code behind it do the interaction with item purchase API.

Owner: Kaidi He

Estimated Time: 1 week

Item Purchase – API Back End

Web API that allows a user to purchase an item for sale. The Web API service will pass the information to the persistence layer which will mark the item as sold in the database.

Owner: Manuel Perez

Estimated Time: 3 days

## User Profile – Front End

Display all the information about a user including his/her current rates. Plus, implement a rating option in this page. The JavaScript code behind it do the interaction with rating API.

Owner: Kaidi He

Estimated Time: 1 week

Rate a User – API Back End

Web API service that allows a user to give a rating to another user after they have purchased an item from them. The API service will process a JSON object containing the rating given to the seller. This rating will be passed to the persistence layer which will update the Items table for the particular product. This will allow an average rating to be generated for a user which will be available as an additional service.

Owner: Manuel Perez

Estimated Time: 1 Week

Search for Items to Buy – API Back End

Web API service that allows a user to search for items to buy. The API will process a JSON object with the particular search parameters passed to the user that can include location, a minimum or maximum price point of an item, and a specific sellers’ name. The API will pass these search parameters to the persistence layer which will query the database for the desired list of items for sale.

Owner: Manuel Perez

Estimated Time: 2 weeks

Find a users’ average rating – API Back End

Web API service that allows a user to view another sellers’ average rating. The API will process a JSON object that contains the sellers’ name. The data will be passed to application logic code which will calculate the users’ average rating based on ratings of every item that a user has sold and been rated on.

Owner: Tingting Chang

Estimated Time: 1 week

Find Top 5 Best Sellers – Back End Application Logic

Back end application logic code that generates a dynamic listing of the top 5 best sellers in the marketplace. This listing will be updated continuously as items are being sold. The calculation will be based on rating, amount sold, and average price. The specific weights of each of those criteria have not yet been determined. This information will be visible on the homepage of the application.

Owner: Tingting Chang

Estimated Time: 2 weeks

Persistence Logic/Database:

User Table:

Set up user table with user first and last name, userID, phone number, email address, and rating. Code DAO layer to update and delete user profile information when necessary.

Owner: Tingting Chang & Michael Rothkopf

Estimated Time: 2 weeks

Items Table:

Set up Items table with item name, price, number items sold, location, and delivery information. Code DAO layer to update and delete Items table as it is modified by users or when items are sold.

Owner: Tingting Chang & Manuel Perez

Estimated Time: 2 weeks

# List of Technology to be used

*List all the technology that you will be using to accomplish this project, including the language choices, libraries or frameworks that you'll be using. Indicate, for what purpose you'll be using the technology and why you chose it. Indicate the pros and cons of your choices.*

Backend Code:

Language Choice: Java

The majority of the group has experience programming in Java and therefore it made the most sense for choosing this language to develop the project.

Frontend Code:

Language: HTML/CSS, JavaScript

They are probably the “only” choice we have to build our user interface in front end. The browser can render these languages directly.

Framework: Bootstrap

The most popular frontend framework in the word. The components it offers can easily fulfill our demand.

Web API:

Framework: DropWizard

Our group is going to use Dropwizard to build the Web API. Since a lot of our group did not have a lot of experience building RESTful web service, dropwizard seemed like a good place to begin due its relatively easy starting point. Dropwizard also includes a number of libraries that help meet the requirements of the project, such as Jackson which can convert Java objects to and from JSON.

Database: MongoDB

Our group has decided to try MongoDB has our database for the project. Since objects are passed from the front end as JSON, we thought it made sense to store this data easily in a document-oriented database. Since our group has more experience using relational databases we still have the option of transitioning back to a MySQL database if learning Mongo proves to be too difficult.