**SCHOOL OF COMPUTING (SoC)**

**DIPLOMA IN INFORMATION TECHNOLOGY**

**ST0503 Back-end Web Development**

**2019/2020 SEMESTER 2**

**ASSIGNMENT 1**

**Instructions and Guidelines:**

1. The assignment **source code** must be submitted before 6th Jan 2020, 9am. You are required to submit your source codes to the BlackBoard. Remember to provide your Class, Group, Admission Number(s) and Name(s) on the softcopy.
2. A one page word document showing the tables you have created and their linkage (including foreign keys) should be included in the submission. ***Done***
3. You are required to **clearly** provide instructions on how to setup the project on the lecturer's laptop in a text file that is to be included in the softcopy submission. **Done**
4. Students are to work in a group of 1-2 members.
5. **Students of 2 member group must complete one of the 2 additional features (without extra marks) stated in the document or be penalized 15 marks**.
6. Marks will be given separately for each student in the group, depending on his contribution to the assignment. The assignment will account for **30%** of your final grade*.*
7. The assignment should be implemented using Node JS, Express and MySQL.
8. The interview will be conducted during the lessons in the week of **6th Jan 2020**. You are expected to explain the program logic and modify the program during the interview. **If you are absent, you will be awarded zero mark for the assignment.**
9. Your application will be tested with POSTMAN.
10. **No marks will be awarded**, if the work is copied or you have allowed others to copy your work. Warning: Plagiarism means passing off as one's own the ideas, works, writings, etc., which belong to another person. In accordance with this definition, you are committing plagiarism if you copy the work of another person and turning it in as your own, even if you would have the permission of that person. Plagiarism is a serious offence and disciplinary action will be taken against you. If you are guilty of plagiarism, you may fail all modules in the semester, or even be liable for expulsion.
11. 50% of the marks will be deducted for assignments that are received within ONE (1) calendar day after the submission deadline. No marks will be given thereafter.

Exceptions to this policy will be given to students with valid LOA on medical or

compassionate grounds. Students will need to inform the lecturer as soon as reasonably possible. Students are not to assume on their own that their deadline has been extended.

# Assignment 1: SnapSell

## Background

You are an aspiring entrepreneur from Singapore Polytechnic on the lookout for the next big business idea. As a poly student, at the start of the semester you would buy books for modules you are taking, and at the end of the semester either toss them out or leave them on the shelf to rot.

You have recently discovered that this was also the case for many of your friends, and thought it was extremely wasteful.

“Wouldn’t it be great if we could sell our used items in a convenient way? That would allow sellers to tidy up their houses, and buyers would be able to snag items at a discounted price. Eureka!”

- You

You found a co-founder whom also related with the pain of tossing away perfectly good books. He is a mobile developer, and you are supposedly an expert backend developer.

The next day, he hands you the API specs for SnapSell.

# Assignment Requirements

You are required to fulfil the following basic requirements:

* Create a new MySQL database with the tables needed for this project. ***Done***
* Proper database design with correct use of primary and foreign key constraints. ***Done***
* Create an Express server that comply with the specs provided. ***Done***
* Consume data from MySQL using the mysql library. ***Done***

Bonus Requirements:

* Create endpoint for image uploading/storage and retrieval of products (with image info) from the server. Server should only accept **jpg** images below **1 MB**. ***Done***
* Create endpoints related to the liking and unliking of a product by another user. Your endpoints should include GET, POST, and DELETE. GET endpoints should provide info of liker and product info including product name and username, which is retrievable by table joins. Note that the appropriate table(s) should be created in the database.***Done***

**(Students of 2 member group must successfully complete one of the above 2 advanced features (without extra marks awarded) or face a penalty of 15 marks. 2 member teams must also provide a listing detailing the contribution of each member. This is used to determine the individual contribution, and members who do not contribute will be PENALIZED).**

**Grading Guidelines**

The assignment will be assessed based on the following criteria:

* Demonstrate and satisfy the web api endpoint functionalities listed below to access/update and return relevant data from the database upon success or failure(75%)

Note: There are 12 APIs, with each api taking 5-10 marks.

Components of grading per api is based primarily on correctness(returning the right json data with proper Model, Controller layer and database calls) and returning the right failure message.

* Proper database design and creation of tables(User, listing and offers) with sample data in MySQL(15%)
* Advanced Features(10%)
* Amount of individual contribution to the project
* Question & Answer during the interview

# SnapSell API Specs

For this assignment, you are required to create the following endpoints. The response message body for failed operations can be determined by you in the API.

## 1)

## Endpoint: GET /users/

**Request body schema**: N/A

### Success Response

**Code**: 200 OK

**Content**: Array of all the users in the database. The user schema below may be different from yours:

[  
 {  
 "id": 1,  
 "username": "steve\_jobs",  
 "profile\_pic\_url": "https://upload.wikimedia.org/wikipedia/commons/thumb/f/f5/Steve\_Jobs\_Headshot\_2010-CROP2.jpg/800px-Steve\_Jobs\_Headshot\_2010-CROP2.jpg",  
 "created\_at": "2019-06-20 18:54:57"  
 },  
 ...  
]

### Error Response(s)

**Condition**: Unknown error

**Code**: 500 Internal Server Error

***Done***

## 2)

## Endpoint: POST /users/

Used to add a new user to the database.

### Success Response

**Code**: 201 Created

**Content**: ID of the newly created user:

{  
 "userID": 1  
}

**Request Body**:

Ensure that the id and created timestamp are autogenerated and not provided by the user.

{  
 "username": "steve\_jobs",  
 "profile\_pic\_url": "https://upload.wikimedia.org/wikipedia/commons/thumb/f/f5/Steve\_Jobs\_Headshot\_2010-CROP2.jpg/800px-Steve\_Jobs\_Headshot\_2010-CROP2.jpg"  
}

### Error Response(s)

**Condition**: Unknown error

**Code**: 500 Internal Server Error

***Done***

## 3)

## Endpoint: GET /users/:id/

Retrieve a single user by their id.

**Request body schema**: N/A

### Success Response

**Code**: 200 OK

**Content**: A single user:

{  
 "id": 1,  
 "username": "steve\_jobs",  
 "profile\_pic\_url": "https://upload.wikimedia.org/wikipedia/commons/thumb/f/f5/Steve\_Jobs\_Headshot\_2010-CROP2.jpg/800px-Steve\_Jobs\_Headshot\_2010-CROP2.jpg",  
 "created\_at": "2019-06-20 18:54:57"  
}

### Error Response(s)

**Condition**: Unknown error

**Code**: 500 Internal Server Error

***Done***

## 4)

## Endpoint: PUT /users/:id/

Update a single user. ID and created timestamp should not be updatable.

**Request body schema**:

Refer to the request body schema for the POST /users/:id endpoint.

### Success Response

**Code**: 204 No Content

**Content**: N/A

### Error Response(s)

**Condition: The new username provided already exists.**

**Code: 422 Unprocessable Entity**

**Condition**: Unknown error

**Code**: 500 Internal Server Error

***Done***

## 5)

## Endpoint: GET /users/:user\_id/listings/

Retrieves all listings of a given user.

**Request body schema**: N/A

### Success Response

**Code**: 200 OK

**Content**: Array of all the listings of a user:

[  
 {  
 "id": 1,  
 "title": "iPhone 6s USED",  
 "description": "In good condition. Camera and screen not working.",  
 "price": 250,  
 "fk\_poster\_id": 1,  
 "created\_at": "2019-06-20 18:54:57"  
 },  
 ...  
]

### Error Response(s)

**Condition**: Unknown error

**Code**: 500 Internal Server Error

***Done***

## 6)

## Endpoint: GET /listings/

Get all listings in the database.

**Request body schema**: N/A

### Success Response

**Code**: 200 OK

**Content**: List of all listings:

[  
 {  
 "id": 1,  
 "title": "iPhone 6s USED",  
 "description": "In good condition. Camera and screen not working.",  
 "price": 250,  
 "fk\_poster\_id": 1,  
 "created\_at": "2019-06-20 18:54:57"  
 },  
 ...  
]

### Error Response(s)

**Condition**: Unknown error

**Code**: 500 Internal Server Error

***Done***

## 7)

## Endpoint: GET /listings/:listing\_id/

Retrieves a single listing by its id.

**Request body schema**: N/A

### Success Response

**Code**: 200 OK

**Content**: A single listing:

{  
 "id": 1,  
 "title": "iPhone 6s USED",  
 "description": "In good condition. Camera and screen not working.",  
 "price": 250,  
 "fk\_poster\_id": 1,  
 "created\_at": "2019-06-20 18:54:57"  
}

### Error Response(s)

**Condition**: Unknown error

**Code**: 500 Internal Server Error

***Done***

## 8)

## Endpoint: POST /listings/

Used to add a new listing to the database.

**Request body schema**:

Ensure that the id and created timestamp are autogenerated and not provided by the user.

{  
 "title": "iPhone 6s USED",  
 "description": "In good condition. Camera and screen not working.",  
 "price": 250,  
 "fk\_poster\_id": 1  
}

### Success Response

**Code**: 201 Created

**Content**: ID of the newly created listing:

{  
 "listingID": 1  
}

### Error Response(s)

**Condition**: Unknown error

**Code**: 500 Internal Server Error

## 9)

## Endpoint: DELETE /listings/:id/

Deletes a listing given it’s id. Idempotent.

**Request body schema**: N/A

### Success Response

**Code**: 204 No Content

**Content**: N/A

### Error Response(s)

**Condition**: Unknown error

**Code**: 500 Internal Server Error

***Done***

## 10)

## Endpoint: PUT /listings/:id/

Updates a listing.

**Request body schema**: Refer to the schema for POST /listings/:id/.

### Success Response

**Code**: 204 No Content

**Content**: N/A

### Error Response(s)

**Condition**: Unknown error

**Code**: 500 Internal Server Error

***Done***

## 11)

## Endpoint: GET /listings/:id/offers/

Retrieves all offers of a listing.

### Success Response

**Code**: 200 OK

**Content**:

[  
 {  
 "id": 1,  
 "offer": 200,  
 "fk\_listing\_id": 1,  
 "fk\_offeror\_id": 3,  
 "created\_at": "2019-06-20 18:54:57"  
 },  
 ...  
]

### Error Response(s)

**Condition**: Unknown error

**Code**: 500 Internal Server Error

***Done***

## 12)

## Endpoint: POST /listings/:id/offers/

Adds an offer for that listing. A user can send more than one offer for the same listing.

**Request body schema**:

Ensure that the id and created timestamp are autogenerated and not provided by the user.

The fk\_listing\_id column should be obtained from the request params.

{  
 "offer": 200,  
 "fk\_offeror\_id": 3  
}

### Success Response

**Code**: 201 Created

**Content**: ID of the newly created listing:

{  
 "offerID": 1  
}

### Error Response(s)

**Condition**: Unknown error

**Code**: 500 Internal Server Error

***Done***