# Master Team Project SoSe2022

## WWW site for Buy/Sell/Share of Digital media exclusively for SFSU students and faculty

## Blue Bird

## GDSD Team 3

## Milestone 2

| Team Member Details | | |
|---|---|---|
| **Deepak Heman Das** | **Team Leader,<br>Frontend Developer** | **deepakrajpal27@gmail.com** |
| **Mutee Ur Rehman** | **Frontend Lead** | **muteeurrehmanbajwa@gmail.com** |
| **Gul Shair Butt** | **Backend Lead** | **segullshairbutt@gmail.com** |
| **Hadil Bader** | **Github Master,<br>Backend Developer** | **hadilbader812@gmail.com** |
| **Trushar Shaileshbhai Mandaviya** | **Backend Developer,<br>Documentation Master** | **trusharmandaviya6@gmail.com** |
| **Sagar Dhaware** | **Frontend Developer** | **sagar.dhaware04@gmail.com** |

## 18.05.2022

## 1. Functional requirements - prioritized

- **Admin Access - must have:**
    1. Every media item uploaded by a user must be approved by the admin first and then go live on the website.

- **Most selling media - opportunistic:**
    1. A logged in user must be able to see the most sold media in the last month.
    2. A logged in user must be able to enable the in-app notifications for most sold media in the last month.

- **Search/Browse function - must have:**
    1. Users must be able to search for media using simple text.
    2. User must be able to filter media based on various key items e-g media type, published_at
    3. Users must be able to view the details of each individual media.

- **Chat Messaging - must have:**
    1. A logged in user must be able to send a message to the seller of a media.
    2. A seller must be able to see the messages received to him.
    3. A seller must be able to reply back to the buyer.

- **My Media - must have:**
    1. Logged in users must be able to publish images, video, music and graphics.
    2. Logged in user must be able to update the media prices, name and description
    3. Logged in user must be able to delete/unpublish the media

- **Download Media - must have:**
    1. A logged in user must be able to download the free media.
    2. A logged in user must be able to buy the paid media.
    3. A logged in user must be able to download the bought media even if it has been deleted/un-published by the owner
- **Editor's Choice - opportunistic:**
    1. Logged in users must be able to see the editor's choice tag on media.
    2. Editor's choice tag must be there using semantic analysis.

- **Image Recognition - desired:**
    1. Image recognition must be implemented to cancel out explicit content on the portal.

## 2. List of main data items and entities

The database will contain the following tables with the following fields:

- **User**: an entity representing a user of the web-based service. A user can be a student/faculty or an admin.
    1. id: a unique identifier of the user
    2. first_name: the user's first name
    3. last_name: the user's last name
    4. email: the user's email address
    5. address: the user's address
    6. is_admin: a boolean value and defines whether the user is an administrator or not
    7. is_enabled: a flag to show whether the user is enabled or not (admin must approve it for it to be enabled)

- **Media**: an entity representing the digital media to be sold, shared, or bought.
    1. id: a unique identifier of the media
    2. title: a title for the media
    3. created_at: date and time when the media was uploaded/created
    4. updated_at: date and time when the media was last modified
    5. description: a description of the media
    6. price: the price if it is paid, default will be 0
    7. is_enabled: a flag to show whether it's enabled or not (admin must approve it for it to be enabled)
    8. owner: user who owns the media
    9. format: type of the media e.g., image, video

- **Order**: an order placed by the user
    1. id: a unique identifier of the order
    2. created_at: the date and time on which the order was created
    3. updated_at: the date and time on which the order was last modified
    4. status: status denoting the status of the order (pending, processing, canceled, or completed)
    5. buyer: user the user that placed the order
    6. cost: the cost of the order
    7. media: media ordered

- **Message**: an entity representing a message sent by one user of the service to another
    1. id: a unique identifier for the message
    2. sender: the user who sent the text
    3. receiver: the receiver of the text
    4. content: the content of the text
    5. sent_at: the time at which this text was sent
    6. seen_at: the time at which this text was seen
    7. status: the status of the message

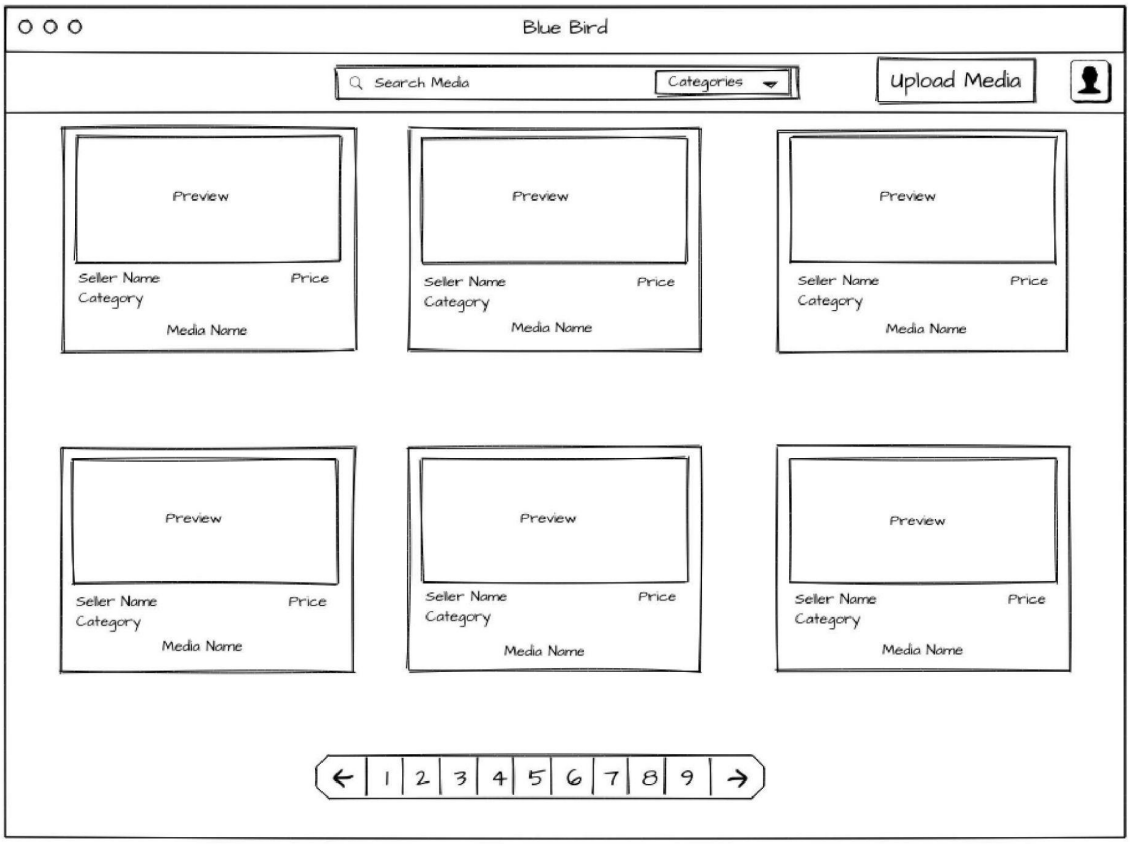
- **Tag**: an entity representing a tag for the media
    1. id: a unique identifier of the tag
    2. name: name of the tag

- **Attachment**: an entity representing a media attachment
    1. id: a unique identifier for the attachment
    2. name: name of the attachment
    3. format: data format of the attachment

## 3. UI Mockups and Storyboards

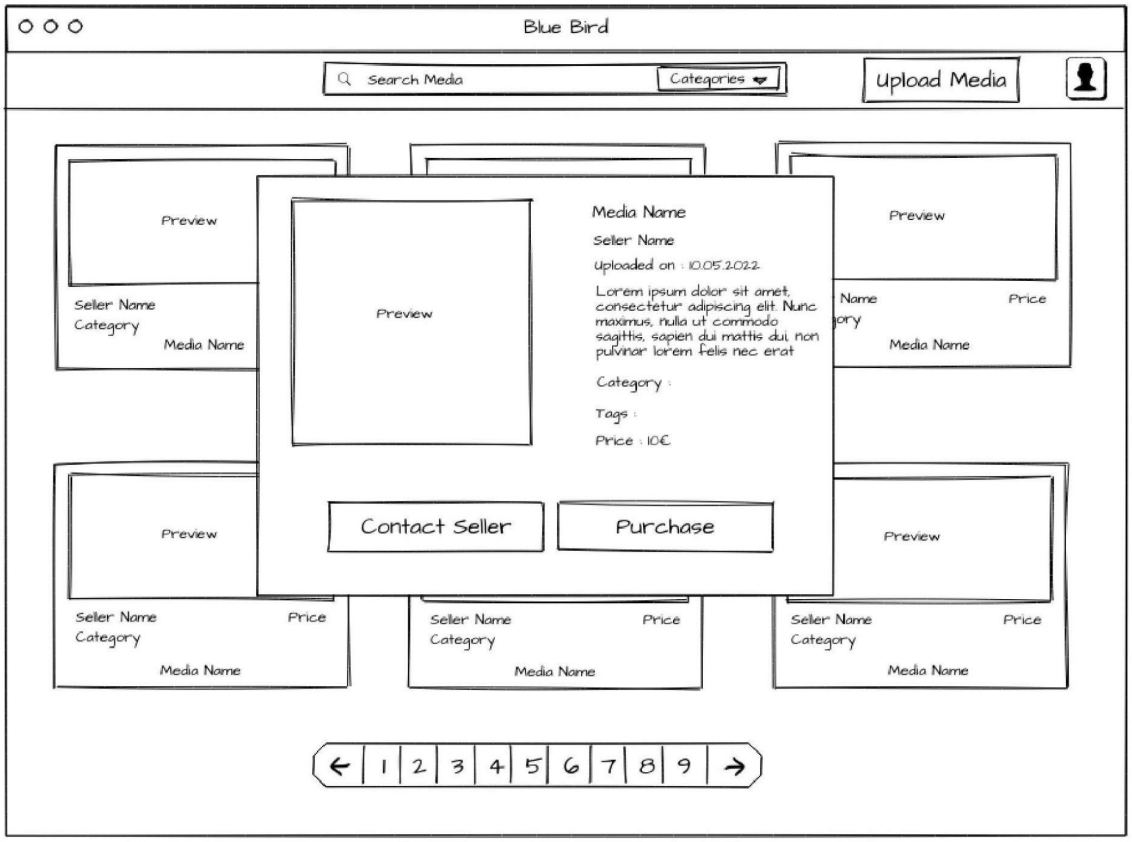
### 1. Home/Landing Screen

Landing screen will have a search field with respect to categories, list of all the items present on the portal, profile button through which user can view/edit profile.



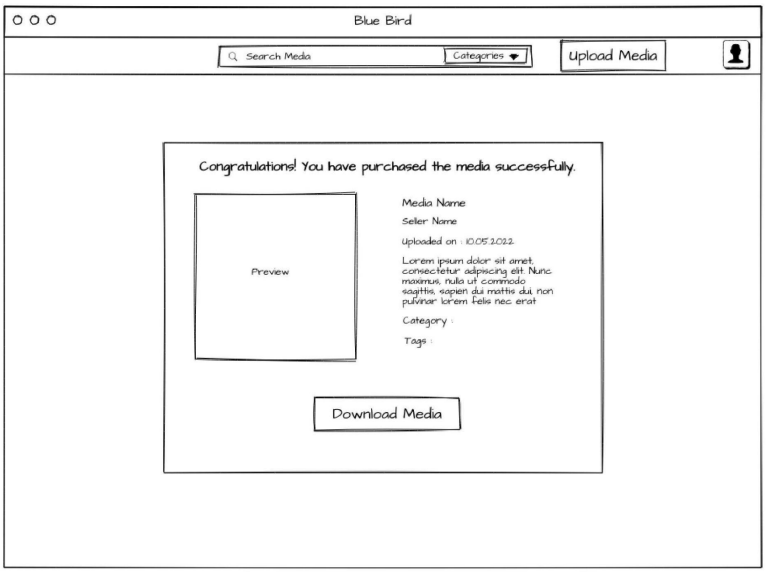

### 2. View Artifact Screen

Here the user can view the artifact details in a dialog box, considering UI/UX dialog boxes are supposed to be a simpler way of presenting artifacts.



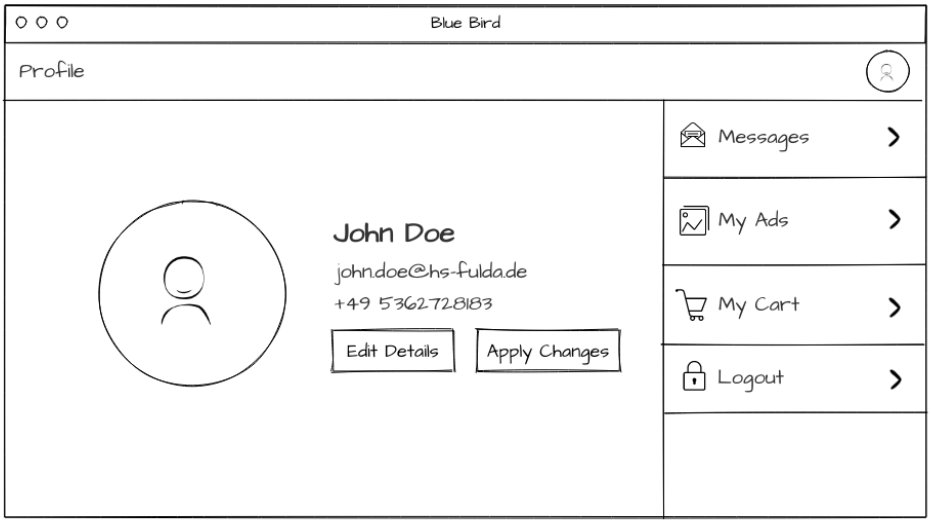

### 3. Purchase Success Screen

On this screen the user will be able to see confirmation message for successful purchase of artifact.



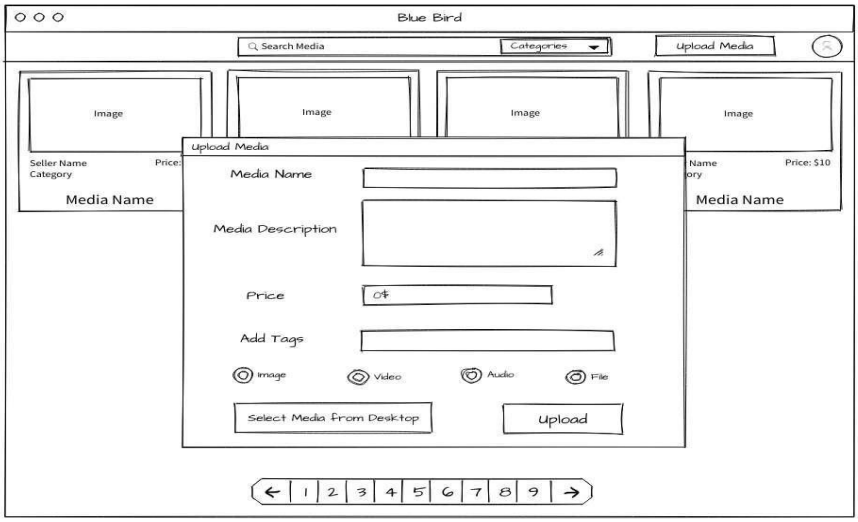
4. **Profile Screen**

The user may modify his or her personal information, and there are options to access to screens such as Messages, Ads, and Cart. From here, a user may also logout.



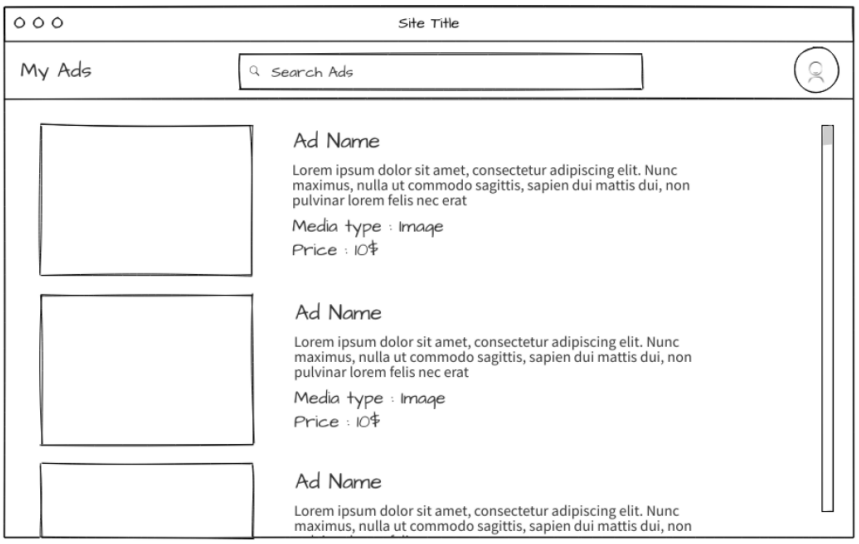

5. **Upload Media Screen**

After pressing the 'Upload Media' button, a modal appears, prompting the user to enter the media's name/title, description, pricing, and type of media.



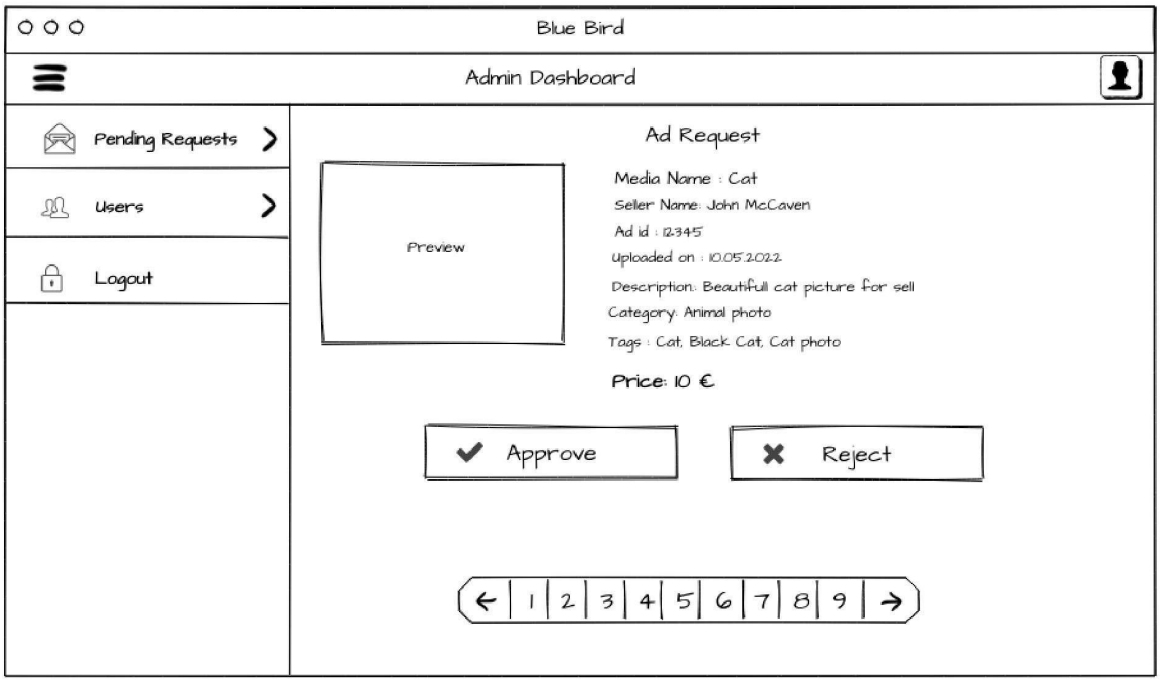

6. **Ads screen**

All user ads along with add details are displayed here. A user can search ads based on ‘title’.



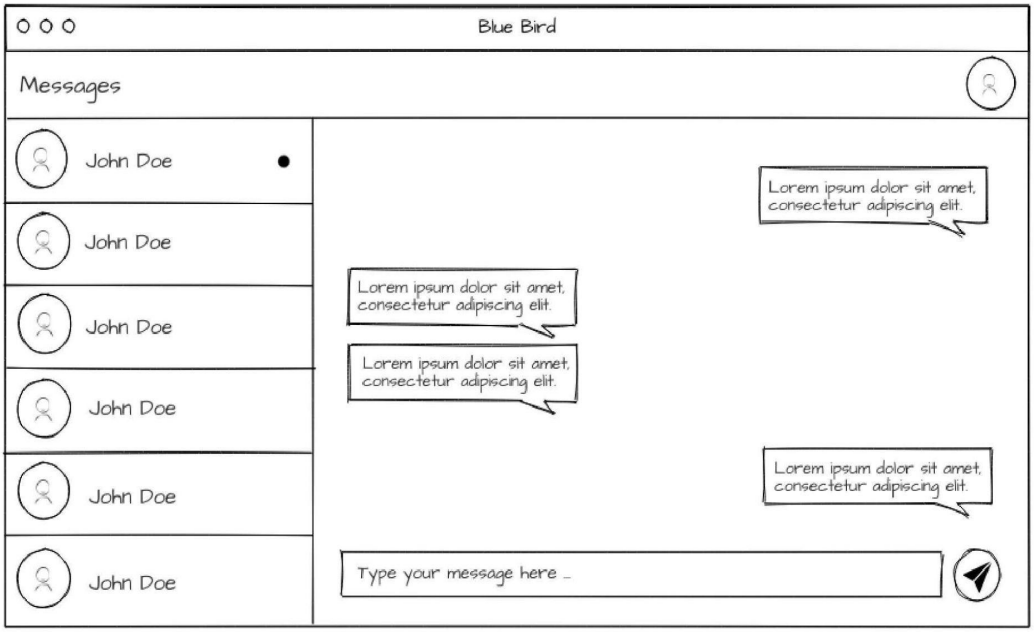

## 7. Admin Dashboard

Admin shall be able to see all the activities by user, admin shall have rights to manually accept or decline media post from user.




## 8. Chat Screen

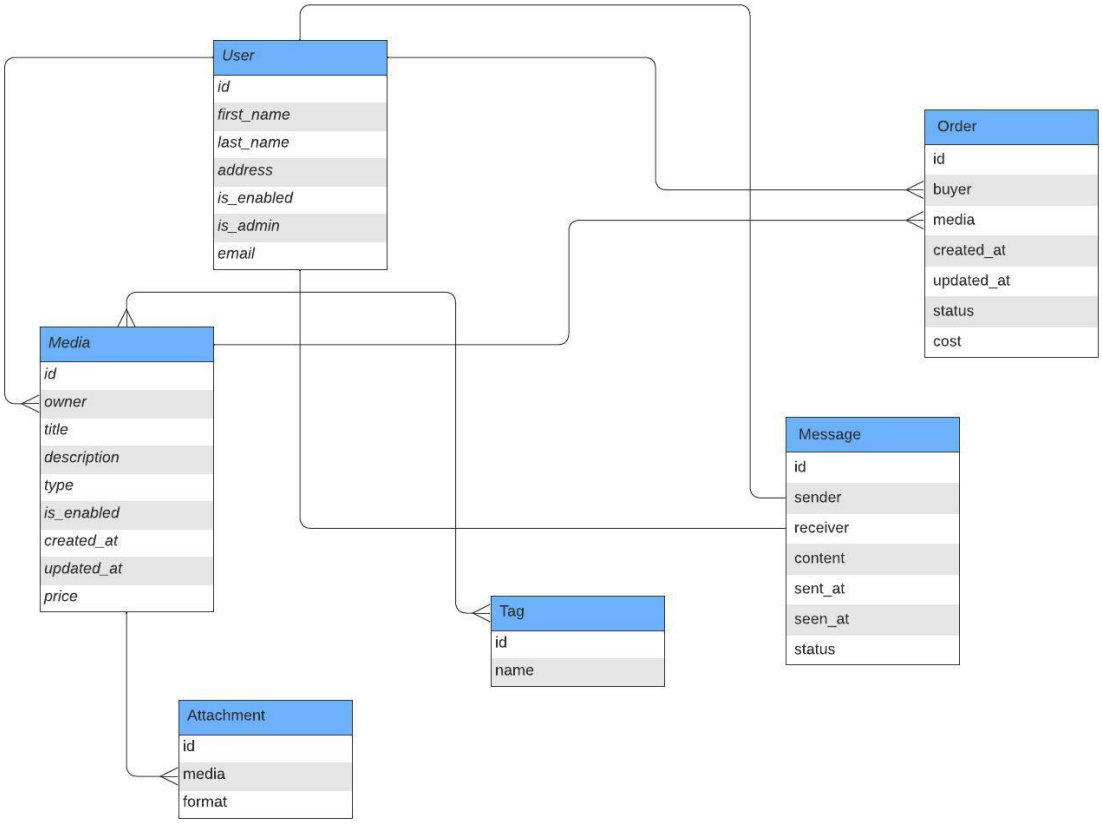
User shall be able to do contact the seller for enquiries he/she have regarding artifact.




# 4. High Level Architecture, Database Organization

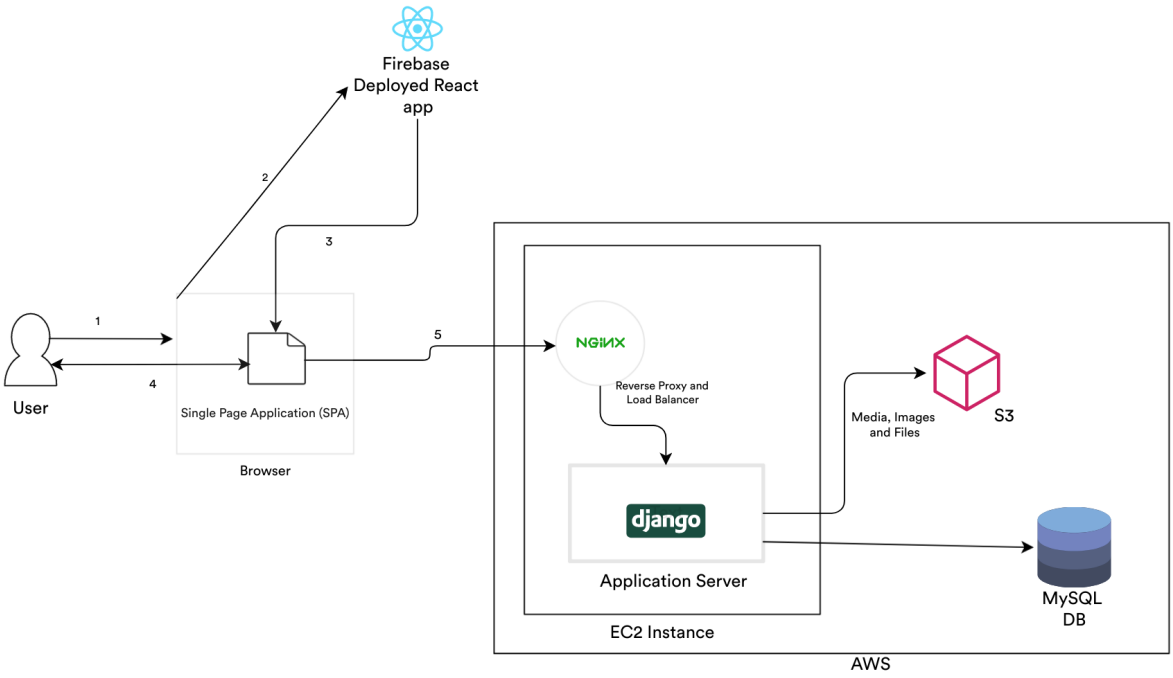
## Database organization

The database will contain the following tables with the following fields and relationships::




## High Level Architecture

1. The application's frontend will be built using the React.js (18.1.0) framework/library.
2. Django (4.0.5) framework will be used for backend development of the application.
3. App will support Chrome, Mozilla, Edge and Safari browsers.
4. Frontend app will be deployed on Firebase Hosting.
5. The Django application will be deployed on an AWS EC2 instance.
6. S3 from AWS will be utilized to store media assets in the cloud, and RDS will be used to host the application's database.
7. Users will interact directly with the frontend, which will connect with the backend via REST apis.
8. Nginx SW installed in the EC2 instance will act as a reverse proxy.




## Media storage:

S3 File Systems storage will be used to store all types of media (files, audio, video, and pictures). Supported media types include:
Media types supported for:

Files: .pdf, .docx, .xlsx,
Images: .png, .jpg, .jpeg
Video: .mp4
Audio: .mp3

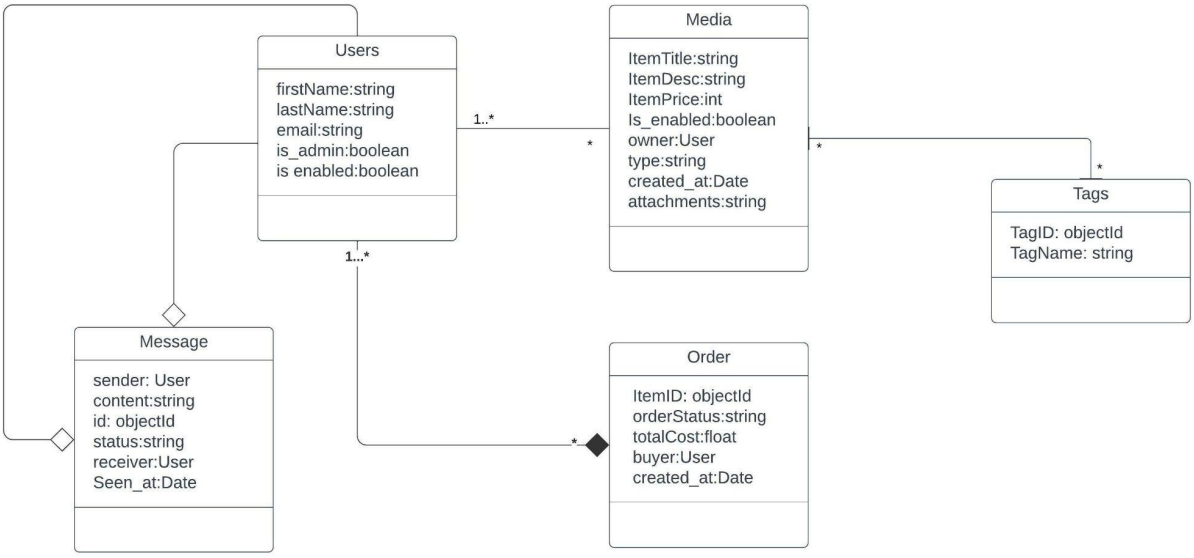
## Search/filter architecture and implementation

The user will be able to freely search for artifacts by media name (DB field: title) and apply filters independently using the ‘Category’ option provided. Filter media by type, such audio, video, picture, or file. Each piece of media will have its own set of tags. Tags can be used to search for media. The system does not supply media tags; instead, the user must generate them. The user may search for media by name, tags, or category, which will be sent to the search API as a query. The landing page will show the results of the search API. Every piece of media will have a 'type' field that will be used to filter content.

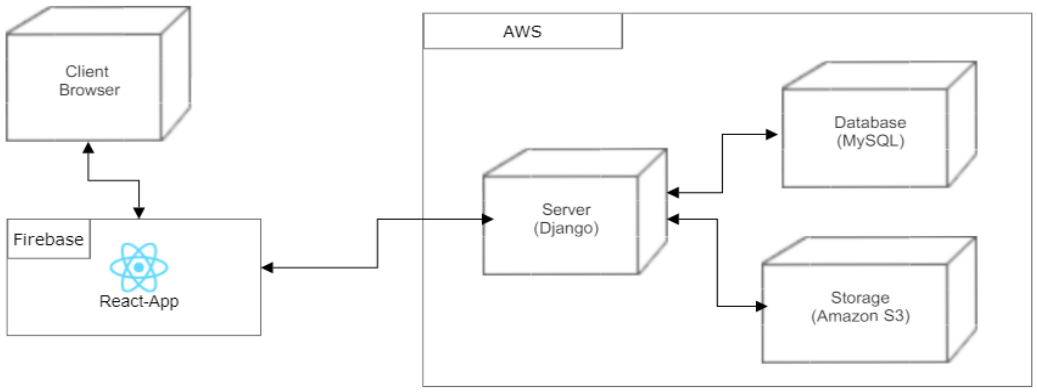
The buyer can rank the vendor on a scale of 1 to 5 after a successful media purchase. The lowest score is one, while the best score is five. The user's overall rating is determined by the average of all such ratings.

# 5. High Level UML Diagrams

## 1. High Level Class Diagram




## 2. Deployment Diagram




# 6. High Level UML Diagrams

### Identify actual key risks for project at this time

1. **Teamwork Risk :**
    a. Deepak, our group leader, is going to Italy in the first week of June. So there could be some risk related to teamwork like contacting the professor for the review of the

milestone document or leading the team in a productive way. But We got good team spirit and everyone is ready to lead the team in the absence of Deepak.

2. **Skill Risk :**
    a. As Trushar is a fresher and he is learning Django from scratch, there could be delay in the development of the backend and ultimately the delivery of the project. But We have Gull Shair who has good experience and is also a backend leader of our project along with Hadil, can guide and accelerate the development pace of the project.

3. **Schedule Risk :**
    a. As most of the team members are taking 7-8 subjects this semester, There are many assignments, submissions and presentations to be finished. So, One could not manage to deliver work on the time assigned to him/her. But We divide tasks and workload equally to each member of the team and we keep track of all tasks so that we do not leave behind the schedule.

## 7. Project Management

For the M2 task we started by dividing all the tasks in frontend and backend categories. These tasks were assigned to frontend and backend teams respectively. We are using Trello to keep track of all the tasks. Tasks like prioritizing functional requirements which require active participation and approval of all teammates were discussed in online meetings.

On Trello we have backlog, todo, on going, done, tested, and deployed stages. Where every assignee is responsible to make his/her tasks on the stage where it currently is. Team lead is responsible for assigning the tasks among the group with the discussion with team members and a tester is being made, who's responsibility will be to test the modules which are in testing phase on Trello.