

Project Report



Decentralised Online Digital Media Marketplace

Submitted on: 25th May 2018

Blockheads Team:

Surya Avala (z5096886)

m: s.avalala@student.unsw.edu.au

Yunhe Gao (z5119451)

m: yunhe.gao@student.unsw.edu.au

Natesh Pai (z3470584)

m: n.pai@student.unsw.edu.au

Scrum Master:

Egene Oletu (z5059371)

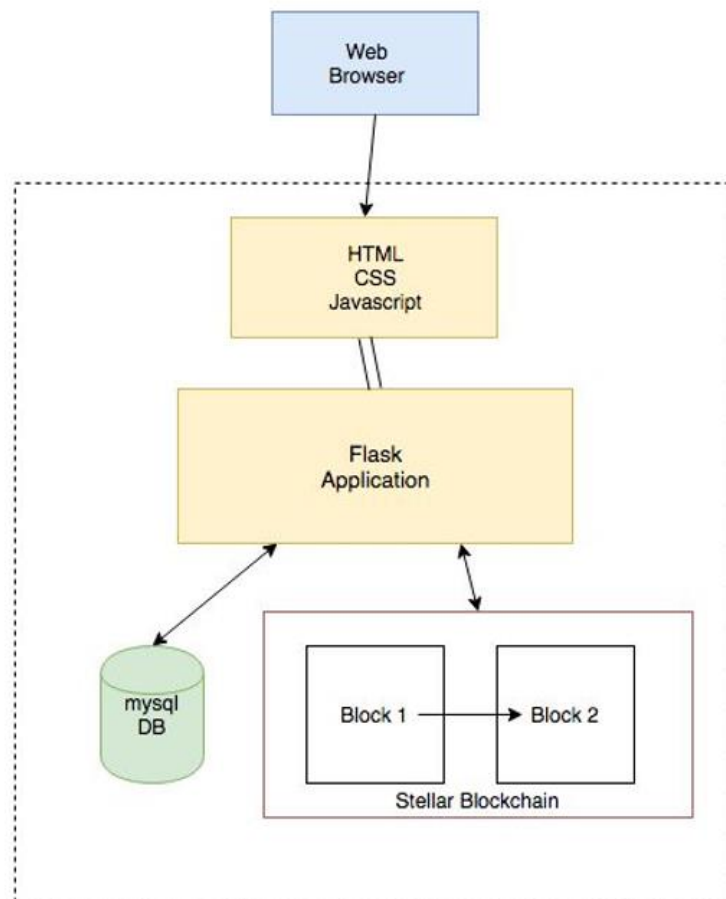
m: e.oletu@student.unsw.edu.au

1. Project Overview:

The project focused on creating a decentralized autonomous content economy, where content value can be recognized efficiently, and all contributors can be incentivized directly and effectively promote long-term economic growth.

The reason companies like YouTube have enormous power over content creators is that they almost monopolize the means of distribution. YouTube, as a privately-owned entity, has its fundamental interest in maximizing profits, which poses an underlying conflict of interest with content creators. The solution is to create a collectively owned, decentralized means of distribution, which ensures all content value is directly distributed to content creators and affiliated contributors without going through a privately-owned entity as a middleman.

2. Background Architecture:



User access the application via their web browser. A web page is generated by the browser using the HTML, CSS, JavaScript code. This HTML, CSS, JavaScript code is generated by our backend Flask application. The Flask application interfaced with the mysql database and the blockchain to achieve appropriate functionality.

2.1 Background on User Interface:

To facilitate the transactions between users for the content, a website was developed that would make the process more intuitive and less harsh. From within the website, users can create an account and sign in. With this account, users can then upload content for purchase by other users, purchase and view and download available content.

The website was built using the flask microframework due to its ease of implementation and suitability to provide a proof-of-concept model of this project. It communicates with the database to collect and store data related to the users and content, and the blockchain, to collect and store data related to transactions.

2.2 Background on Database:

Database is an important way to manage data set, and support information that are necessary in program processing to solve real world management problems ('database' 2018). This concept provided an effective way to store, manage and collect data for analysing or processing purpose. The purpose of using database in this application is just to validate login and to display data in the front end. In this project, major functionalities such as user creation, video purchasing and retrieving transaction history are implemented with the help of blockchain. The database is only used to store basic information like user details and videos. There are mainly four tables in the application. They are "Table_Comments", "Table_Purchases", "Table_Users" and "Table_Videos". The user table is used to store user login credentials and the keys which are used for blockchain transaction. Video table is used to storage attributes of video. "Table_Comments" and "Table_Purchases" are used to keep the information related to customer comments on videos and the video that are being purchased by a particular user(s). The only purpose for these two tables are to make our website more efficient and responsive.

2.3 Background on Blockchain:

The blockchain (in our case Stellar Test Network) is used to primarily record User details, Asset details (media item details) and Transaction details, but not the actual media storage itself. A blockchain module (in Python3) was written on top of existing core libraries to interface with the Network. The flask backend calls on the blockchain module to communicate with the blockchain.

3. Application Features with Process Flow

The main feature of this project is that it allows users to buy and sell content, in particular mp4 videos. To facilitate this transaction between users, a graphical user interface was developed, in the form of a website, that would make this process, from the initial upload of content, to the purchase of said content, more intuitive to users. In brief, the current implementation has the following features:

- Create an account
- Sign into account
- Browse available content
- Search for specific content using keywords
- Upload content
- Download content
- Purchase content
- Playback content
- View transaction history

The website was built using the flask microframework due to its ease of implementation and suitability to provide a proof-of-concept model of this project. Flask breaks up the website into a series of routes, with each route being related to a specific URL. At each route in the flask app, access to specific pages can be controlled so that certain features can only be accessed by registered users. For example, uploading content is a feature that only registered users have access to. Data from forms is also processed at each route and used to trigger certain events such as URL redirects or user creation in the database and blockchain. And finally, since each route is related to a specific URL, most if not all routes are responsible for serving up relevant pages. Each page is written as a Jinja2 template to allow for dynamic features.

Since the flask app is capable of receiving and changing data in the system, a suitable way of dynamically changing the websites pages was needed to reflect these changes. For example, when a user uploads content, this new content then needs to be accessible to all other users. To achieve this, the Jinja2 templating language was used. Jinja2 is templating language for python that uses documents written in a combination of html and python to create dynamic webpages. These templates get rendered using data provided by the flask server to generate complete html pages interpretable by modern web browsers.

Following are the primary logic flows interfacing with the blockchain:

i. User Signups:

Every time a user signs up on to our platform using the front end the following blockchain related logic/smart contract gets triggered:

- a. A random Mnemonic string gets generated in the [BIP:39](#) standard.
- b. A public, private key pair gets generated using the Mnemonic string from part a.
- c. The Mnemonic string a.k.a passphrase is stored in the Database in order to be able to retrieve if the user forgets it.

- d. A user account is then created on the blockchain with the public key. Once created, it can be accessed/verified by anyone using the following url https://horizon-testnet.stellar.org/accounts/<public_key>, where <public_key> is the public key for the user.

ii. Asset creation:

Anytime a user uploads a media item into our platform the following blockchain logic/smart contract gets triggered:

- a. A new custom asset (currency) is created with that represents this media item.
- b. The user signs a multi-signature escrow to be able to transfer this asset to other users.

iii. Transactions:

Every time a user (user1) purchases an item from another user (user2) the following blockchain related logic/smart contract gets triggered:

- a. Balance and asset ownership checks are performed to make sure that
 - i. User1 actually owns the asset
 - ii. User2 has sufficient balance to purchase the asset
- b. User2 signs the multi-signature escrow (already signed by User1 in step “a” of asset creation logic), to be able to accept that custom asset.
- c. Then User1 sends a piece of his asset to User2 in exchange for “Lumens” (native currency on Stellar block chain) that are quoted as the price for the media item.
- d. Transaction “c” is then signed by the “app admin” as an extra layer of security.
- e. A string of data (less than 28bytes) is included in the data section of the transaction indicating that the transaction on the blockchain is referring to the particular media item that was purchased.
- f. All these transactions are recorded on the blockchain and are verifiable by anyone by looking at their account pages. (and also shown in our app)

4. Project Deviation and Reasoning

Most of the function that are listed in our project proposal have been attained. However, there are few deviations in our final product compared to the initial proposal. The scrum team had decided to use Ethereum blockchain which eventually changed to Stellar Network as Ethereum is a public network and needs real-time funds to transact. The team used private network called Stellar and used XLM tokens to perform transactions. Also, the project did not focus much on building a friendly user interface, rather it focused much on developing a flexible and core functionality. The main intension behind this is, the team believed functionalities, blockchain and shopping cart was the primary focus. Hence the UI kept changing throughout the project based on functionality implementation. This was one of the drawbacks as

team missed to recognize UI was equally important. The second deviation from the initial plan was that the team decided to develop a statistical analysis and transaction history page as the team wanted to exhibit more of the blockchain in real-time scenario, but this led to increased response time. The team also spent significant time retrieving the information from blockchain to display transaction history but succeeded well at the end.

Thus, to conclude, in addition to the initial proposal, the team has deviated with blockchain platform and developed two more pages/ functionalities. The statistic page displays a line chart that shows the transaction count for each day corresponding to a particular user. This data is fetched from user's purchase history in blockchain. In the transaction history page, every transaction that has been completed by the users are listed.

5. Project Implementation

The Project was implemented using the SCRUM methodology. The team consisted of Scrum Master, Product owner and developers. Across the duration of 12 weeks of project implementation, the stakeholders carried out several tasks in different phases as below.

5.1 Initiation Phase

The first two week were spent forming the scrum team, deciding and identifying the Scrum master and the product owner based on everyone's area of expertise. The team explored different topics and based on each team members interest, the concept of blockchain was adopted in the project. Later, the team came up with an idea of selling and buying digital media through a decentralized application of an Ethereum blockchain. However, the blockchain was changed from Ethereum to Stellar considering the pros and cons on each platform.

The team worked on deciding epics for the project, and thus came up with 5 epics which were later redefined, prioritized and detailed that were disclosed in the project proposal. These epics helped the team to prioritize and draw the initial project backlogs to begin with. The user stories were also overviewed.

5.2 Planning and Estimation

During this phase the stakeholders decided on the project timeframe, estimates, backlog tasks, sprint backlogs and sprint meetings. The applications such as bitbucket for code repository, Trello for tracking project backlogs and sprints and slack for communication was adopted.

The team decided on having two sprint meetings in a week to discuss about the individual sprint tasks and overall status of the project. Every member of the scrum team created their sprint backlog consisting of the tasks to be executed in the sprints.

5.3 Implementation

This phase consisted of execution of the activities and the tasks that were created or decided in the sprint planning meetings. Each member of the team contributed toward the project by working on the areas that were assigned to them by the Scrum Master. The backlogs, sprint backlogs and the code repositories were updated timely by the team members. The different sprint stages were set on Trello which was followed during the project implementation. In every sprint meeting, the status of in-progress sprint and the sprint backlog was discussed.

Overall, this phase consisted of three progress demos to all the stakeholders, and several sprint meetings within the team and the execution of assigned tasks.

5.4 Review and Retrospect Phase

While approaching the end of implementation phase, the scrum team started reviewing the work and project features through project demos and internal demonstrations to the team members. This led to review the implemented or planned sprints. The backlogs were reviewed and updated in this phase. The team also ended up adding additional features of displaying transactions and statistics that were retrieved from the blockchain. The sprint meetings in the review phase also helped the team to get each other's approval on the individual tasks and thus update the status of the deliverables created/declared in the sprint.

During the retrospection phase, the team discussed the ideas and issues that were faced by each member throughout the sprint. The outcomes were utilized positively by learning and enhancing the product and planning the future sprint.

5.5 Final Demo and Release

In this phase of the project, the team emphasized on presenting and delivering the product with agreed functionalities to customer/ primary stakeholders and with this, the team identifies, collects and documents the details of different project deviations and issues faced during the entire duration of the project.

During the final demonstration, the primary stakeholder and the Scrum team members joined together to retrospect the product. This helped to identify the areas of improvement to be implemented in future release, perform a bug-fix on final product and thus release/ submit the project to the primary stakeholder.

6. Project Interface

Setup

To start, unzip the provided project.zip to the destination of your choosing.

To run this application, python 3 and the packages listed in the requirements.txt file will be needed. Assuming that both python3 and pip3 have already been installed, the required packages can be installed using the command:

```
pip3 install -r requirements.txt
```

from the root folder.

After everything has been installed, run the following command from the root directory to start the web server locally:

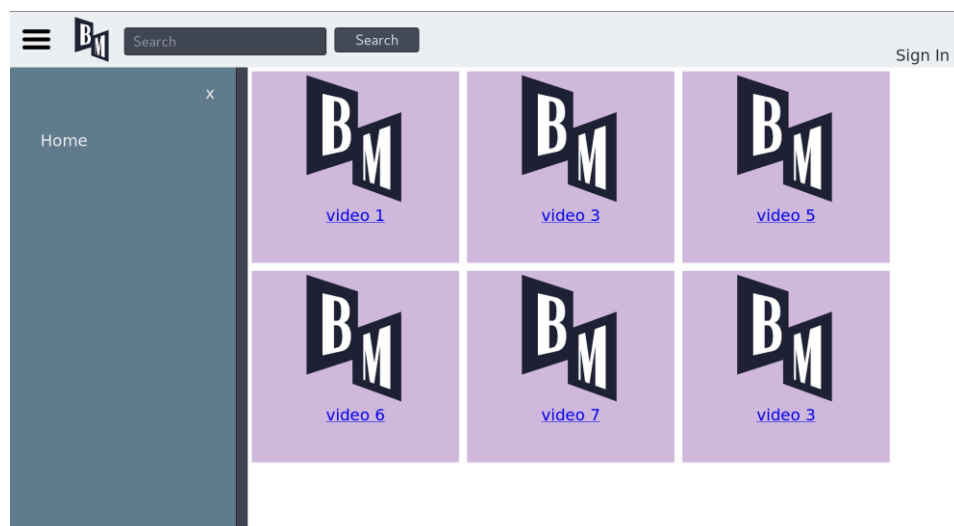
```
python3 run.py
```

To access the website, go to the following URL with a web browser:

```
localhost:3900
```

Using the website

The aforementioned URL will load the index page as shown below. This page will show a list of available videos (shown in purple), provide a form for searching for a video in the top navigation bar, and link to the sign in page.



To sign in to your account click the sign in button in the top right hand corner of the screen. This will load the page shown below

Sign In

Remember me? ☐

Sign In

New to Blockheads? [Create an account](#)

If you have an account already, you can enter in your email and password in this form and click sign in to log into your account. If you don't already have an account, you can create one by clicking on the "Create an account" link show above in blue text. This will open up the page shown below.

Sign Up

*

*

*

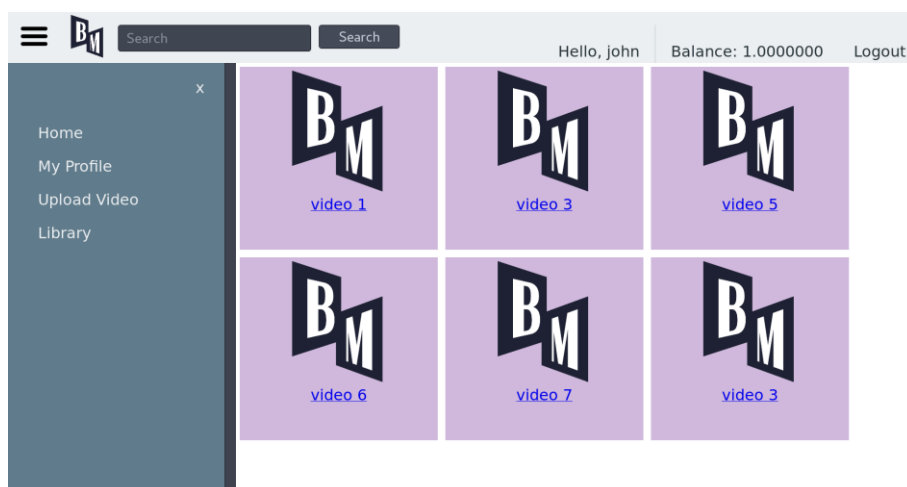
*

*

* required fields

Create account

By filling in each field and clicking "Create account", your new account will be created, assuming the email provided isn't already in use, and sign you in. After signing in on the sign in page, or after creating an account, you will be directed back to the index page.



If you click on the menu icon (3 horizontal lines) in the top left hand corner of the screen, you will open the side navigation menu, as shown in the above image. To upload a video, click on the "Upload Video" link in the menu. This will take you to the page shown below.

BM

Search

Search

Hello, john

Balance: 1.0000000

Logout

Upload a Video

Title

Description

Price

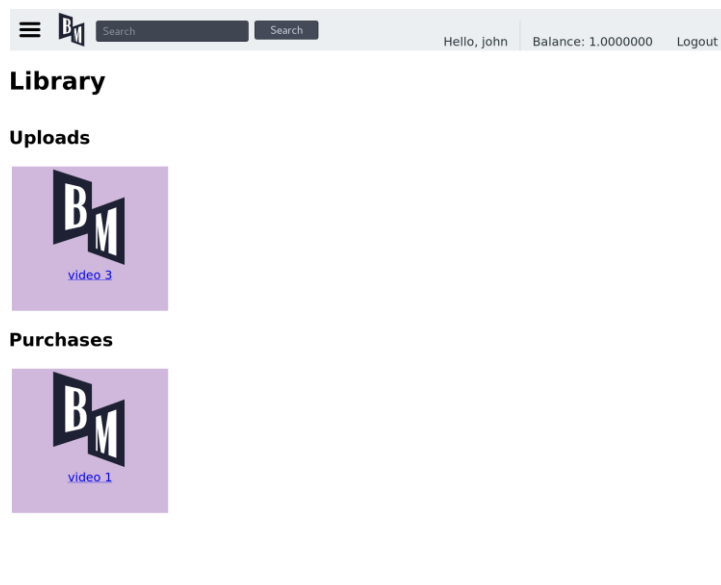
Labels

Browse...

No file selected.

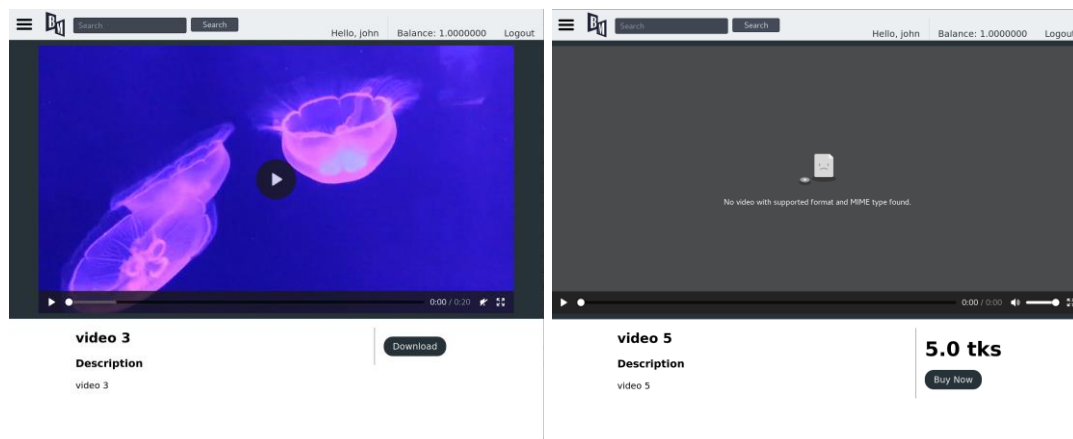
Upload

After filling in the fields shown in the above form, including uploading an mp4 file via the “browse” button, you can upload the video by clicking the “Upload” button. This will take you back to the index page where you should see your uploaded video added to the list of videos on the page.

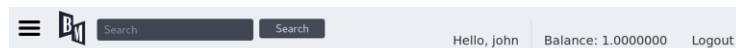


You can also see a list of your uploaded videos, as well as your purchases in the library page as shown in the above picture. This page is accessible by the “Library” link located in the side navigation menu.

To watch a video, click on the tile of the video you want on the index, library, or search pages to bring up a page like one of the pictures shown below



The picture on the left shown what the page would look like if you own or have purchased the video. Videos that are owned or purchased are able to be played by clicking the play button in the centre of the video player, or downloaded by clicking on the download button, below and to the right of the video player. If you do not own or have not purchased the video, then you will see the page on the right. Video playback is not possible from here and the video will need to be purchased before playing or downloading the video is possible. To purchase the video, click on the “Buy Now” button below and to the right of the video player. This will open up the page below




Clicking on the cancel button will stop the transaction from continuing and take you back to the video page. Clicking confirm will confirm your purchase and the system will try and perform the purchase for you. If successful, you will be redirected to the video page and be able to play and download the video.

7. Resources and Third-Party tools

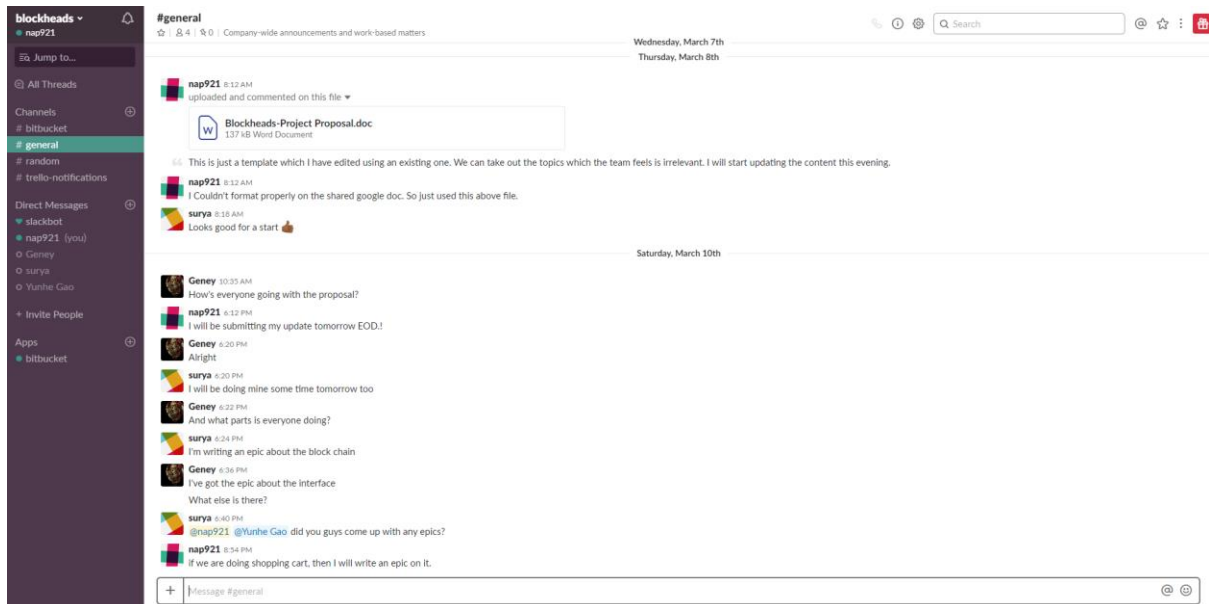
The Scrum team consisted of four members as follows:

- Eugene Oletu: Scrum Master – Project Manager, UI Developer and Primary driver of the project
- Surya Avala: Product Owner – Product planner and Blockchain Developer
- Natesh Krishna Pai: Blockchain Developer, UX Designer and UAT
- Yunhe Gao: Database Developer and Statistical designer

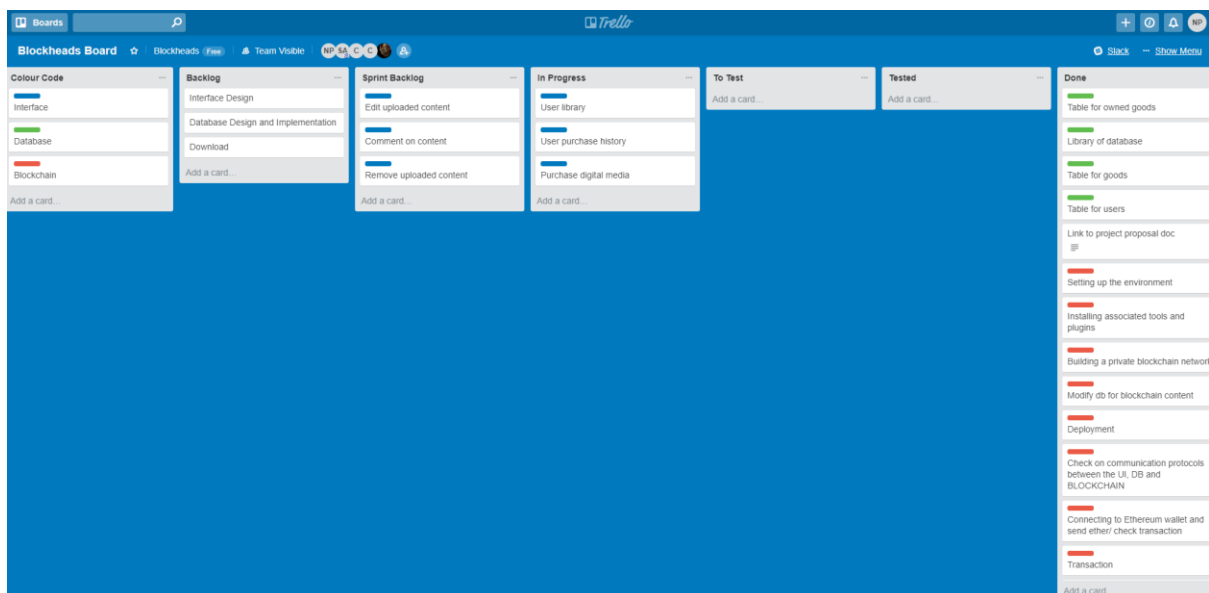
During the entire course of project, the scrum team utilized several third-party tools such as Slack for effective communication, Bitbucket for maintaining code-repository and Trello for project management.

 **Slack:** The slack tool helped the team members to communicate effectively and share the documents and updates with ease. This kept the

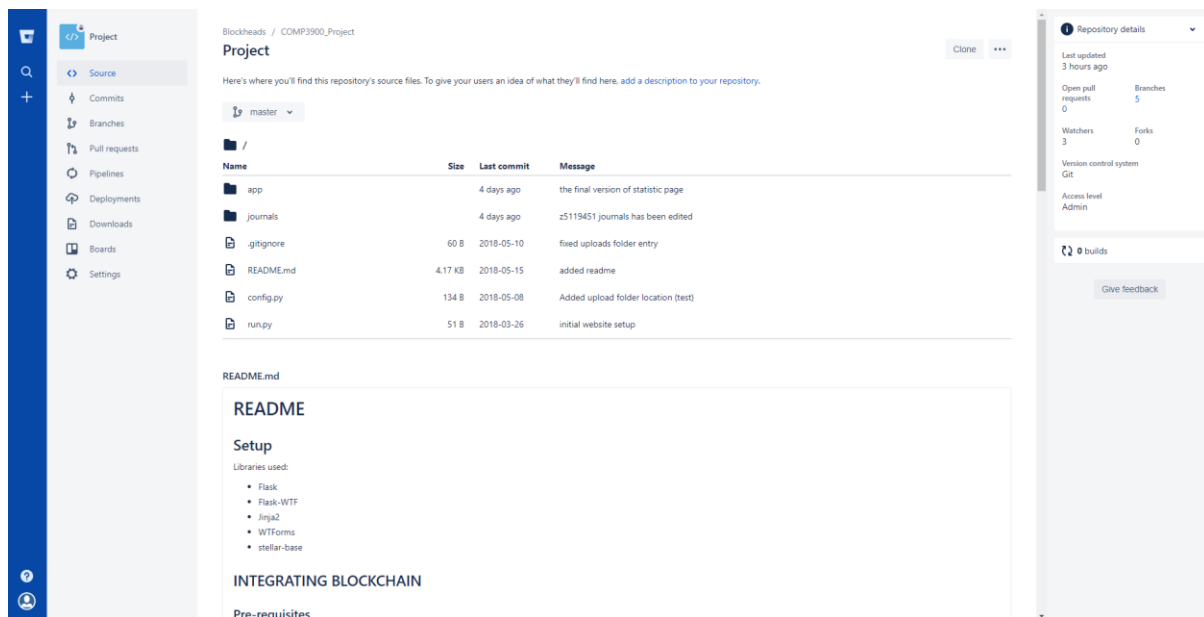
project going very well as there was no communication gaps. Slack also supported direct feeds from Trello and Bitbucket, thus helping the team members to be up-to-date with the recent activities of project as well as each team member.



Trello: The management of the entire project was handled through Trello. It helped the team to keep track of the product backlogs, sprint backlog, in-progress tasks, In-testing tasks, bugs and completed tasks. The colour code functionality of the application helped us to identify each member's sprint with ease.



- **Bitbucket:** This code-repository and version controlling tool played the major act in development, collaboration and real-time handling of the project developments. The different branches were used by the developers to update their work which was then finally merged to the master branch after testing. This way, all the Scrum team members had the updated version of the entire application. The team also utilized Bitbucket to keep track of their weekly work by maintaining journals that was available to view for all members and primary stakeholders.



8. Future scope

A decentralised file distribution system like “Interplanetary File System” (IPFS) could be explored as a possibility (to store the media assets), to make the application truly decentralised in every sense.

An additional user-friendly webpage (unlike the ones that were provided above) where the user can verify/look at their accounts, balances and transactions on the blockchain without having to trust our application.