TEXAS A&M UNIVERSITY

Team 3GB - BitBid
Project Report

Table of Contents

Summary

The stakeholder is Prof. Korok Roy from Mays Business School. The primary customer requirement is to build an auction website **BitBid** where buyers and sellers make transactions in bitcoin. The prototype however indicates that the user interface and payment settlement algorithm are of top priority. The client expected a website that settles with virtual money. The Bitcoin settlement feature was out of the scope of this project's duration. To this extent, the project BitBid that we implemented offers virtual money exchanges between buyers and sellers. It implements the all-pay auction design with the money routing between all the participants in an auction. However, since we were ahead of our delivery timeline, we additionally facilitated bitcoin payments through Coinbase API. While the bitcoin payment feature has not been thoroughly tested due to time and budget constraints, the rudimentary version is ready and could be extended for further development. We were in frequent communication with the client and with their constant feedback, we, **(3GB) BitBid** delivered the functionality that was in scope for the duration of the project and implemented an additional feature too.

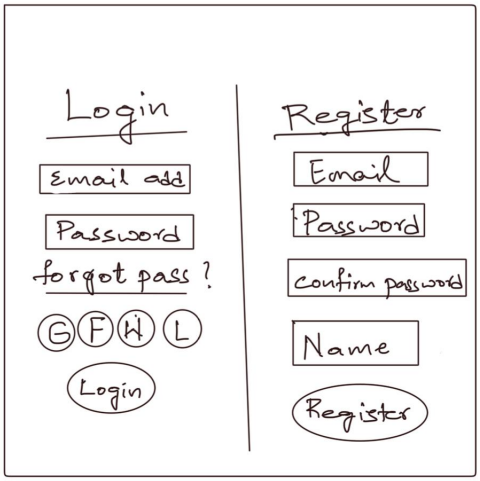
The features delivered in this project are Authentication, The Auction Block, Auction Status, Seller interface, Buyer interface, Payments and Settlement. We used Google authentication provider to make the login easier for users. The Auction block allows the users to view the active auctions on the website. Every auction has a close date and also displays the current status, highest bidder, and corresponding bid of the auction. The Seller interface allows the seller to start and set the terms of a new auction. The Buyer's interface allows a buyer to bid on the items listed on the auction block. The buyers can bid multiple times, whereas the sellers are not allowed to bid. Every user is also provided with an option to add money to their wallet. There are two ways of doing so. They can either add virtual money or use the Coinbase API to top up their wallet. Whenever an auction's horizon (active period) ends, it is processed for settlements. Every buyer pays their bid regardless of whether they win. The seller gets to keep all the bids money. By implementing all these features, we were able to meet all the deliverables set by the client. The future scope for this project would be to integrate different auction models, adding 3rd party wallet support, and use cron jobs to settle the auction.

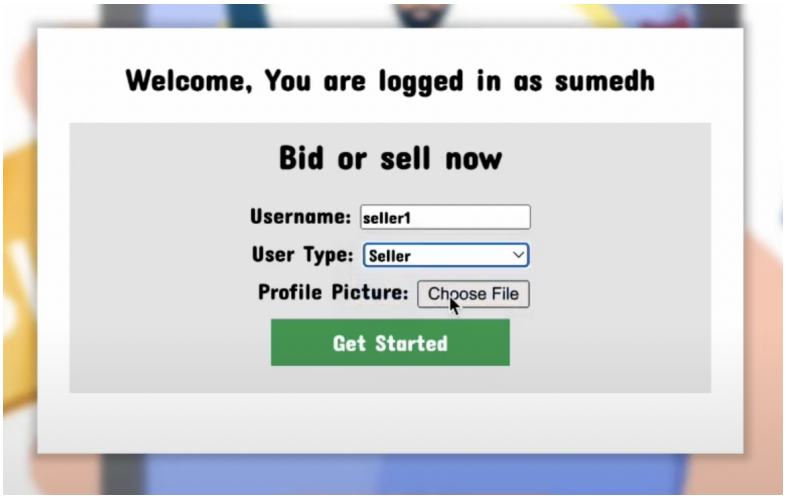
User Stories

Ability to register As a new general user So that I can log in I want to create an account by registering on BitBid	**3 Points**	**Status:** Accepted

According to the initial design document shared by the customer, we used Google Authentication for the users to register on our website. The first time a user logs into the website, he is prompted to log in via his Google Account. If he is a first-time user, we provide him an additional registration form to get other details like username, profile picture, and if he chooses to be a buyer or a seller.

The initial mock-up for the registration page and the actual interface are as follows:

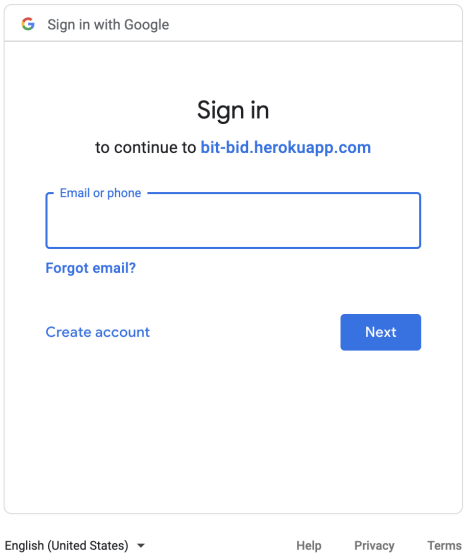




Ability to login As a general already registered user So that I can view items to bid I want to log in and start using the website	**3 Points**	**Status:** Accepted

As described in the previous story, this story included working on generating the Google API key and integrating Google Authentication into our project.

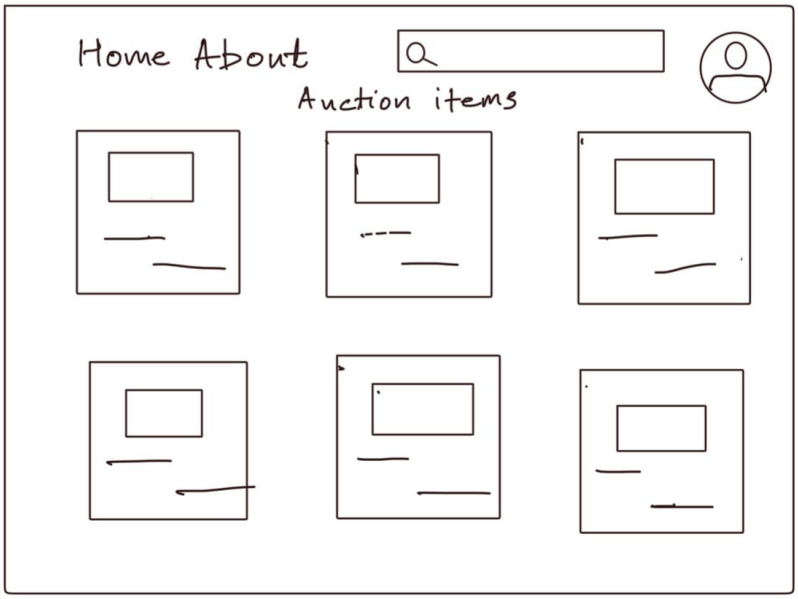
Below is a screenshot of the Google Login View:



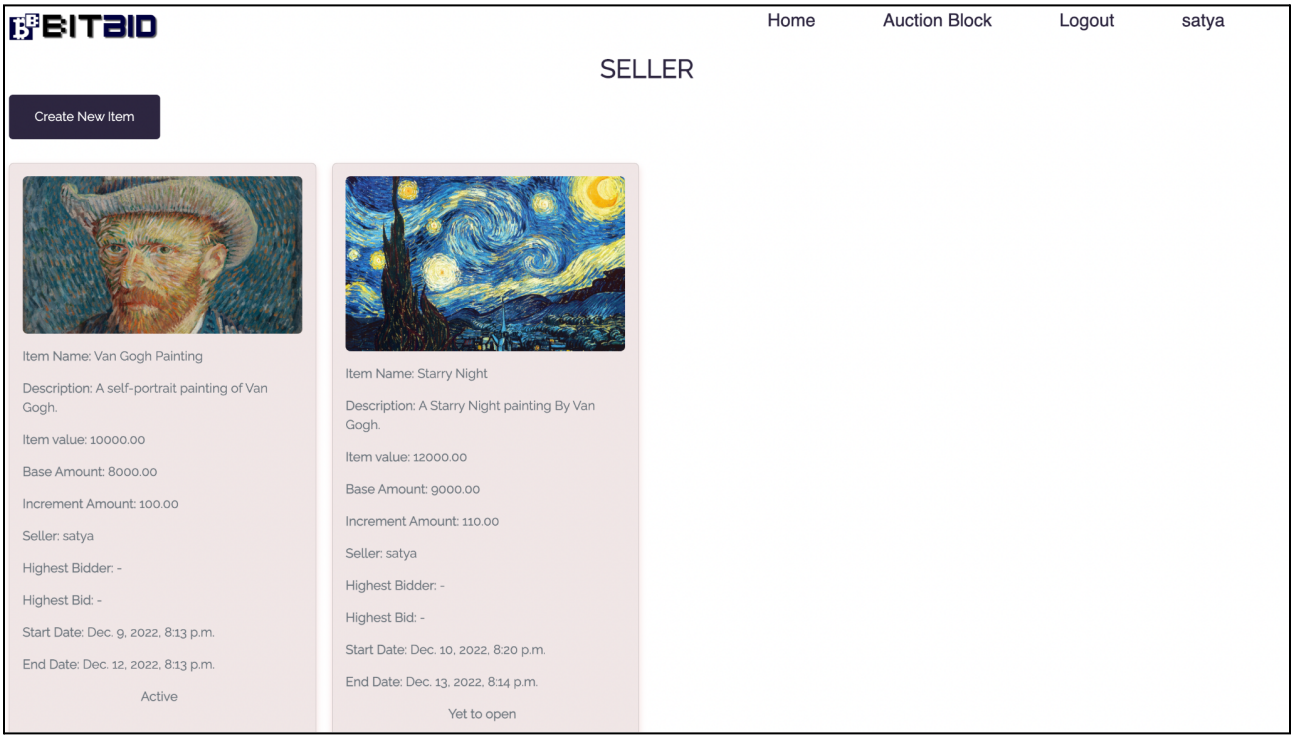
View the items As a logged-in user So that I can choose items to bid on I want to view the items	**3 Points**	**Status:** Accepted

A user will be able to view the items up for auction in the **Auction Block.** A buyer can also see upcoming items on the block before the auction begins. A seller would also be able to see what other sellers are auctioning to get a better idea of the items being sold.

Below is an initial mock-up of the view:



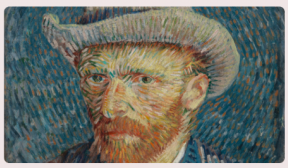
The seller view is as follows:



The buyer view is as follows:

BITBID Home Auction Block Logout pravija

BUYER



Item Name: Van Gogh Painting

Description: A self-portrait painting of Van Gogh.

Base Amount: 8000.00

Increment Amount: 100.00

Seller: satya

Highest Bidder: -

Highest Bid: -

Start Date: Dec. 9, 2022, 8:13 p.m.

End Date: Dec. 12, 2022, 8:13 p.m.

Active

Bid

Coming Soon ...

Item Name: Starry Night

Description: A Starry Night painting By Van Gogh.

Base Amount: 9000.00

Increment Amount: 110.00

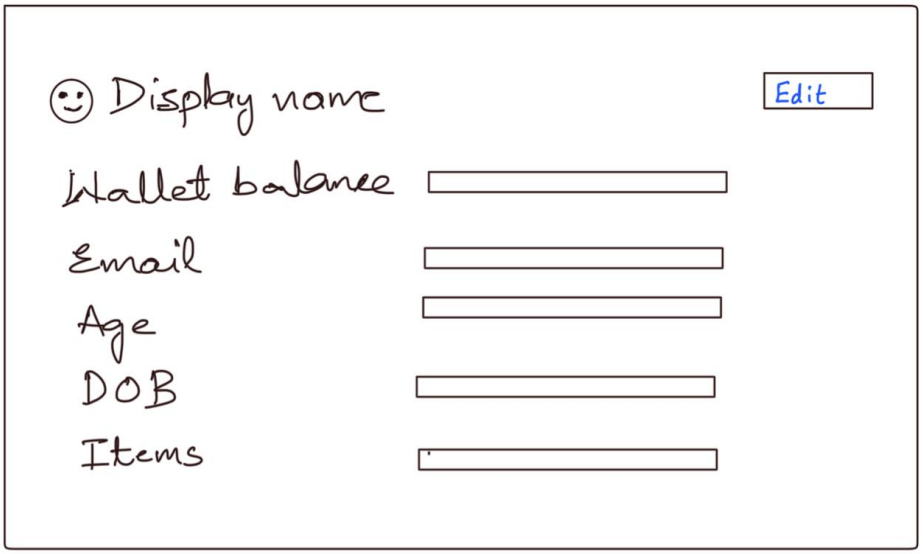
Seller: satya

Start Date: Dec. 10, 2022, 8:20 p.m.

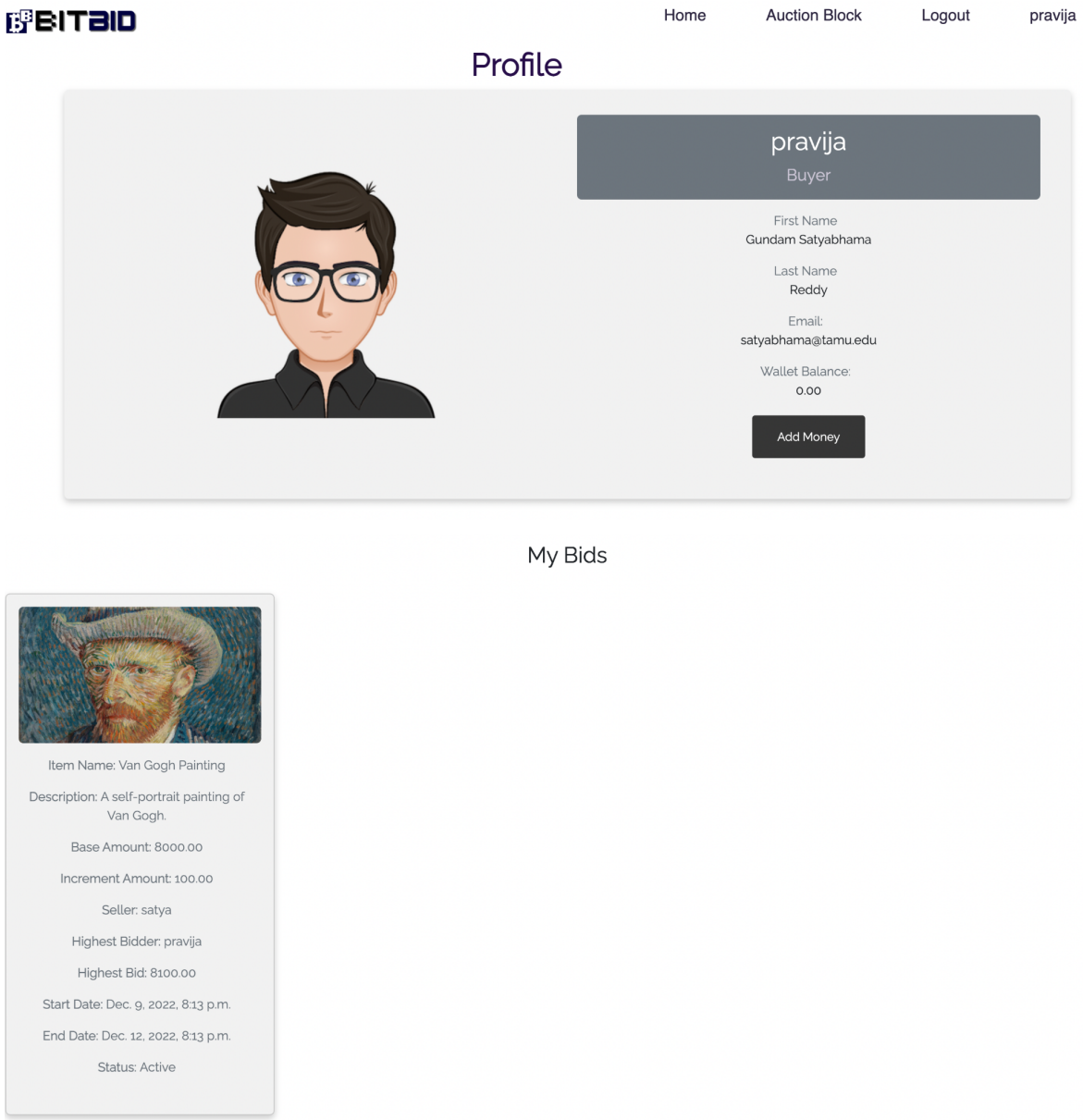
End Date: Dec. 13, 2022, 8:14 p.m.

Yet to open

Differentiate between buyer/seller As a user who has logged in for the first time ever when I login I should be redirected to a page where I can choose to be a buyer/seller from a dropdown.	**3 Points**	**Status:** Accepted
Once the user is logged in, he will be redirected to his respective view based on if he is registered as a seller or a buyer. A new user, however, is prompted with the registration form as described above.		

View User Profile and Wallet Balance As a logged-in user I want to see the bitcoin money balance that I have on the profile page so that I can buy items.	**3 Points**	**Status:** Accepted
A profile page displaying the user information and his wallet balance was implemented. In addition, he can also view the items he has bid on previously here. Below is an initial mockup of the profile page: 		

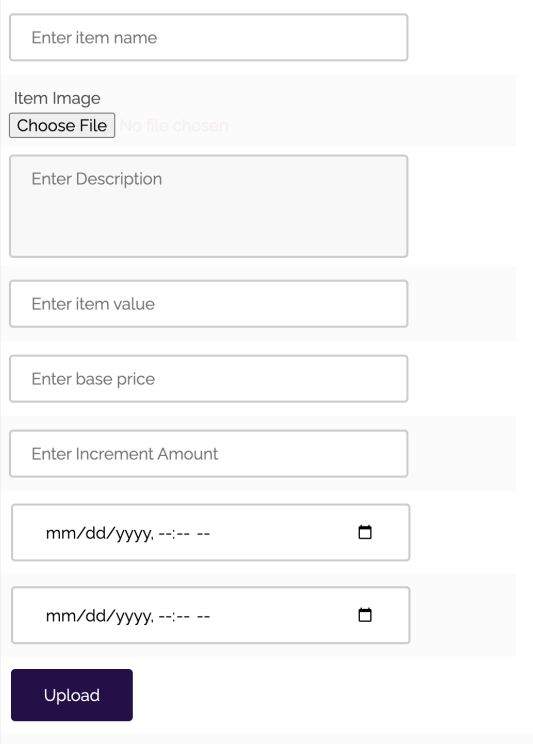
And here's the implemented profile page:



Set the terms for the auction of an item As an admin(seller) I want to set the parameters for the auction to start so that users(buyers) can view the items for auction.	**3 Points**	**Status:** Accepted

A seller can set the terms for the auction of an item using a form. The requirements for the item fields were discussed with the client and redesigned during the course of the project.
The initial mockup for setting the terms of the auction looked like this:

The implemented view is as follows:



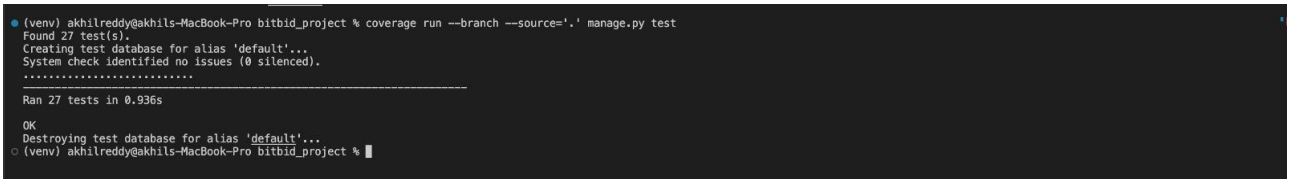
Bids Settlement As a buyer, if I bid on an item, my wallet balance should be debited by the bid amount, while the admin(seller) wallet balance should be incremented by the same amount	**3 Points**	**Status:** Accepted
Once the auction has ended, the bids are settled asynchronously and the wallet balances are updated based on the settlement logic. Since this is an all-pay auction, all the payments done by the buyers are credited to the seller's wallet. A future improvement for this feature would be to notify the users once the settlement is finalized.		

Handling edge cases of settlement As a seller, I may specify an item's start date and end date in my time zone, and the buyer must be able to view these dates in their time zone. Managing instances in which there are no buyers.	**3 Points**	**Status:** Accepted
The settlement logic described in the previous story, has a lot of edge cases to be handled. For instance, if an item has no bidders, then the settlement logic must not fail.		

BDD tests As a developer, I want to execute automation scenarios So that, I can verify that UI is functional.	**3 Points**	**Status:** Accepted
As part of this story, a few more additional BDD tests were added and existing test failures were fixed. The framework used to implement these tests is python's **pytest** module.		

Test coverage As a developer, I want unit tests and code coverage metrics. So that, I can make sure future development does not break existing functionality	**3 Points**	**Status:** Accepted

As part of this story, unit tests were implemented for all the features and a test coverage report was generated. Python's **coverage** module was leveraged to achieve the same.
Below is a screenshot of the unit tests:



And the corresponding coverage report:

```
Name                                Stmts   Miss Branch BrPart  Cover
-------------------------------------------------------------------
bitbid_project/forms.py                 7      0      0      0   100%
bitbid_project/models.py               72      3     10      2    94%
bitbid_project/settlement.py           35     20      8      1    42%
bitbid_project/urls.py                  8      0      0      0   100%
bitbid_project/views.py               129     26     28      2    80%
-------------------------------------------------------------------
TOTAL                                 251     49     46      5    79%
```

UI - rebrand Enrich and fix CSS across the whole application	**3 Points**	**Status:** Accepted
The focus until this story was on the proper feature implementation and the correct functionality. Hence, this story was taken up to improve the user interface of the website.		

Payments As a user, I want payment functionality So that I can add real money to my wallet	**3 Points**	**Status:** Accepted
As per the initial design document, we implemented our functionalities using fake money. As a part of this story, we integrated Coinbase API to add cryptocurrency into user wallet. The risks accepted are as follows: 1. Coinbase api takes the journey out of the application. 2. The app provides the option to add real money (FIAT) instead of bitcoins directly. The conversion from FIAT to CRYPTO is taken care of by coinbase. 3. The payments may not be instantaneous. Coinbase presents the user with a payment link that is valid for up to 60 minutes. Once the payment completes, coinbase calls an API provided by the app by the url `/webhook`. 4. The wallet is updated only when Coinbase calls our webhook api. Naturally any fault with coinbase becomes an accepted risk.		

Adding Images As a User, I want the images feature. So that, I can distinguish between different users and different items	**3 Points**	**Status:** Accepted
Adding, storing, and displaying the item images and profile pictures of users were handled in this story.		

Responsiveness As a User, I should be able to access the website from any device.	**3 Points**	**Status:** Accepted
As many of the users also access the website through their mobile devices, we took up this task to make the website responsive across all devices.		

Automated build image Automated build process to eliminate the dependence on developers	**Status:** Accepted
We also implemented an automated build process to make the deployment process smoother. Instead of manually building the docker image, we made use of github's workflow to automatically build the image and trigger the heroku release. Any push into release branches will trigger this workflow. The build status can be monitored directly from github. The heroku login api credentials are set in the github repository's secrets (please refer to the github's documentation on workflows).	

Team Members and Roles

Team Members:

- Sumedh Basarkod sumedhpb@tamu.edu (Product Owner)
- Akhil Chilumuru akhilchilumuru@tamu.edu
- Rithin Pullela rithin@tamu.edu
- Pravija Danda pravija123@tamu.edu
- Jayashree Mallipudi jayashree@tamu.edu
- Gundam Satyabhama Reddy, satyabhama@tamu.edu

Sumedh Basarkod acted as a product owner throughout the project development while the rest of the team members took up the role of scrum master for each iteration. In each iteration, the entire team was involved in the development process. We followed the handover process, in which at the end of each iteration scrum master discussed the current sprint overview as well as the backlog items (which are ready to be taken up in the following sprint) with the next scrum master and the product owner. In this way, it sets the expectations for the upcoming sprint. After the customer meeting, changes to the sprint stories (if any) were made.

Iteration Accomplishments

Iteration 0:

Based on the client discussion ,we have created the following user stories:

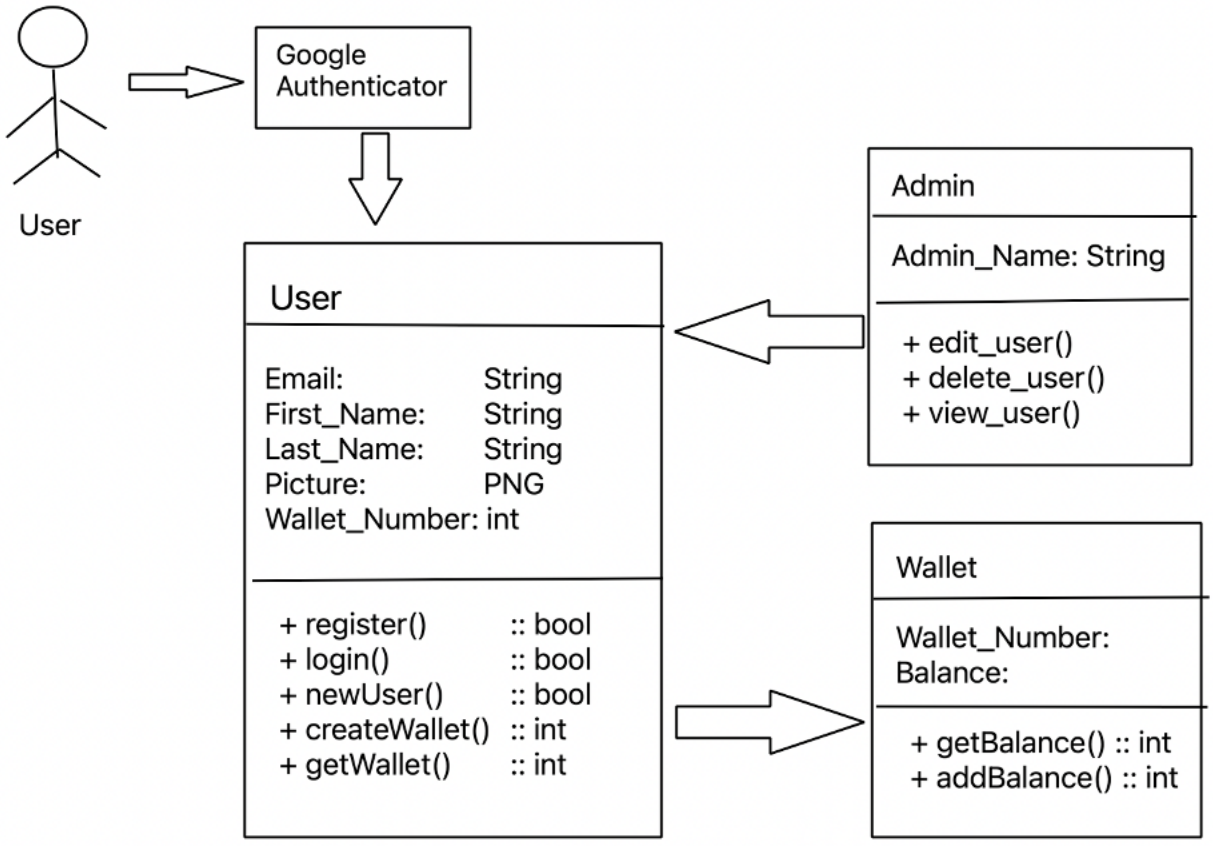
- Ability to register
- Ability to login
- View the items
- Set the terms for the auction of an item
- See the wallet balance

Iteration 1:

Based on our discussion with the client, the following user stories were initially planned to be delivered by the end of iteration 1 and we were able to complete them successfully.

- Ability to login (3 points)
- Ability to register (3 points)

We have scheduled another weekly call with the client to have feedback on the project. As per the request of the client, we kicked things off with a “Google Authentication” Login feature development. We have developed the requested features in a behavior-driven design (BDD) approach. We created the relevant test case for the happy path. The sad path is actually taken care of by Google's Authentication API. The whole project is deployed to Heroku and both the user stories are tested. Also attached below is the Design Diagram for the feature. We have scheduled a meeting with the client on the 17th of October to get comprehensive feedback from the customer.

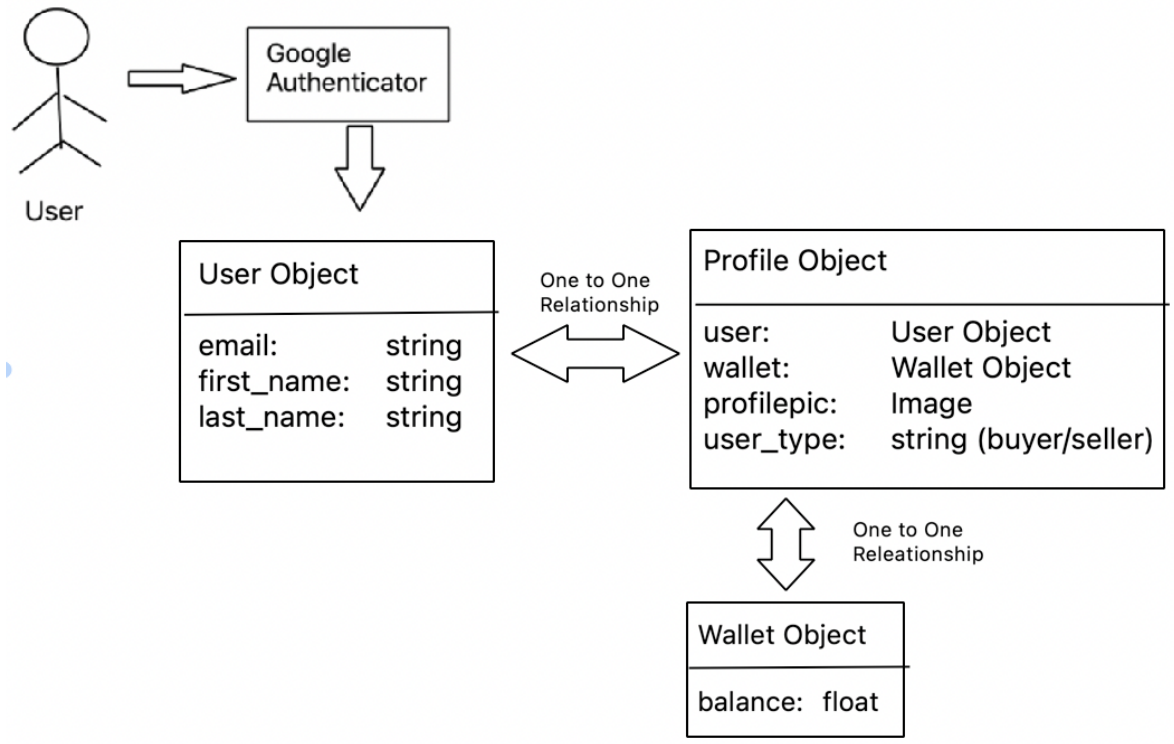


Iteration 2:

We met with the client on our weekly scheduled call Monday at 6 pm and gave the demo of iteration 1. Based on our discussion with the client, the following user stories were initially planned to be delivered by the end of iteration 1 and we were able to complete them successfully.

- Differentiate between Buyers and Sellers(3 points)
- View User Profile and Wallet Balance (3 points)

From the discussions with the client, it was understood that the "Buyer" and "Seller" type of users must have separate views. We concluded to develop the "differentiation" of the users along with a few minor UI refreshments for iteration 2. We changed the structures of the models to accommodate the "user type" variable. The changes are represented in the Design Diagram below. Along with "Google Authentication" we ask the user to select the user type and thus separate the views of both types of users. We used BDD and TDD to develop these stories and deployed them to Heroku. We met with the client on Monday, 24th October, and received positive feedback.

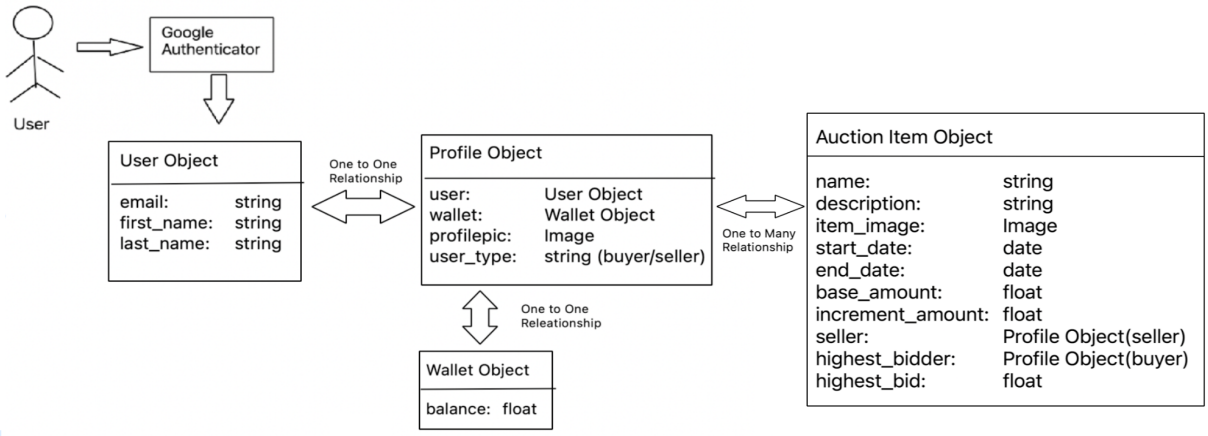


Iteration 3:

We spoke to Josephine (jogonzalespa@gmail.com) who represents the client on behalf of Prof. Korok Ray . We met with the client on our weekly scheduled call Monday, November 7th, and gave the demo of iteration. Based on our discussion with the client, the following user stories were initially planned to be delivered by the end of iteration 3 and we were able to complete them successfully.

- Set the terms for the auction of an item (3 points)
- View the auction items (3 points)

We created interfaces for the buyers and sellers and implemented functionalities in accordance with the client's needs. The client was satisfied with the functionalities provided by the app. We have discussed further the settlement of the auction and wallet management. The development added the functionality for the seller to add new items to the auction. The buyer can bid for the items and the highest bid and bidder details will be displayed in the auction block. The changes are represented in the Design Diagram below. We used BDD and TDD to develop these stories and deployed them to Heroku.

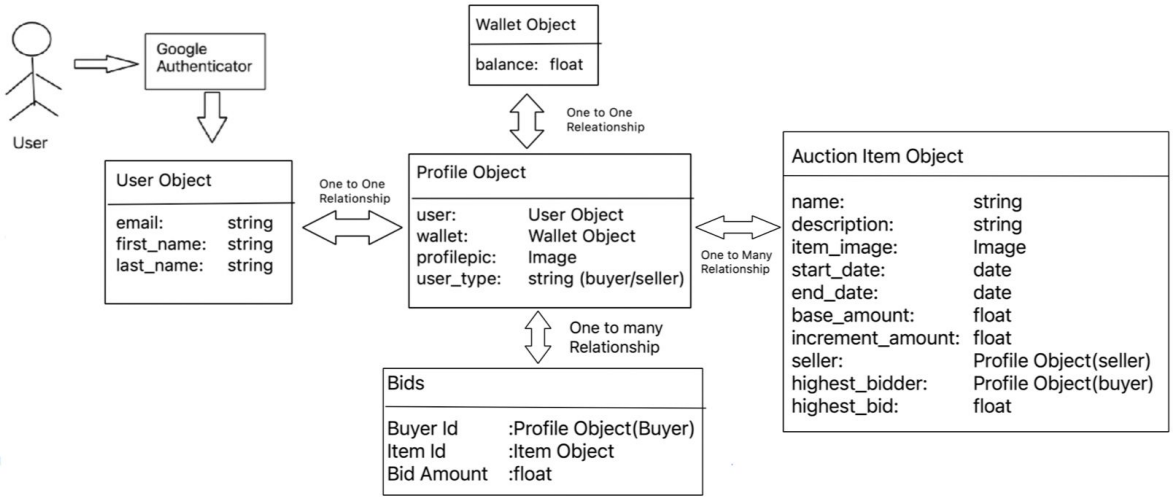


Iteration 4:

We met with the client on our weekly scheduled call on Monday, November 14th, and gave the iteration demo.Based on our discussion with the client, the following user stories were initially planned to be delivered by the end of iteration 3 and we were able to complete them successfully.

- Set the terms of the Bids Settlement (3 points)
- Handling edge cases of settlement (3 points)

We created a settlement of the auction functionality and wallet management for the buyers and sellers and implemented functionalities in accordance with the client's needs. The client was satisfied with the functionalities provided by the app. We have discussed further the implementation of being able to handle different time zones for all the buyers. The development added the complete functionality for the buyer to bid on the auction items. The buyer can bid for the items and the highest bid and bidder details will be displayed in the auction block and the wallet will also be updated accordingly. The changes are represented in the Design Diagram below. We used BDD and TDD to develop these stories and deployed them to Heroku.



Iteration 5:

We met with the client on our weekly scheduled call on Monday, November 21st. The client wanted additional functionality in terms of bitcoin payments. They also requested improvements in the UI interface. We picked the following stories for this iteration.

- Implementing payment using bitcoins (3 Points)
- UI Rebrand (3 Points)
- BDD Tests(3 Points)
- Code coverage(3 Points)
- Resolution of website(3 Points)
- Displaying Images on the website(3 Points)

We have discussed the difficulty of testing the bitcoin payment feature. We used the Coinbase API to facilitate payments.
After integrating with coinbase API

1. We are able to add real money (FIAT) instead of bitcoins directly.
2. The payments may not be instantaneous. payment link is valid for up to 60 minutes.
3. The wallet is updated only when Coinbase calls our webhook API.

There were no UML changes at this iteration.

Customer Meetings

Point of contact:

The client meetings were held with Josephine (**jogonzalespa@gmail.com**) who represents the client on behalf of Prof. Korok Ray . The meetings were held every Monday since October 7th. The recorded meetings can be found here.

Meetings Held:

1. **21 September,2022**
   We Met Prof. Korok Ray at Schlotsky's, off College and University Ave, on 21st September 2022 at 12:30pm. From the discussions with the client, we have understood the design of the website and we have requested for the design document.

2. **7 October,2022**
   Since October 7th, the meetings were held with Josephine on behalf of client, we discussed with her about the design requirements and got started implementing the login feature.

3. **17 October,2022 at 7:30 am**
   By this client meeting, we got the official design document from them and we gave them the demo of the user login feature using google authentication as per the client's requirements.
   From this client discussion, it was understood that the “Buyer” and “Seller” type of users must have separate views,so we completed it by our next client meeting.

4. **24 October,2022 at 7:30 am**
   We developed the feature of the “differentiation” of the users to differentiate between buyers and sellers along with a few minor UI refreshments

5. **7 November,2022 at 7:30 am**
   We have implemented buyer and seller interfaces and given the demo in this meeting and we further discussed about the settlement of the auction and wallet.

6. **14 November,2022 at 7:30 am**
   We have given the demo on the settlement functionality in this meeting and The development added the complete functionality for the buyer to bid on the auction items. The buyer can bid for the items and the highest bid and bidder details will be displayed in the auction block and the wallet will also be updated accordingly. and We have discussed further the implementation of being able to handle different time zones for all the buyers.

7. **21 November,2022 at 7:30 am**
   We have given the complete demo of the settlement of auction items between sellers and buyers at different timezones and the client wanted additional functionality in terms of bitcoin payments.They also requested improvements in UI.

8. **28 November,2022 at 7:30 am**
   We have given the complete website demo after the required UI enhancements and integrating with coin base bitcoin payments functionality.

9. **5 December,2022 at 7:30 am**
   A final demonstration of the entire website was presented and the feedback received was positive.

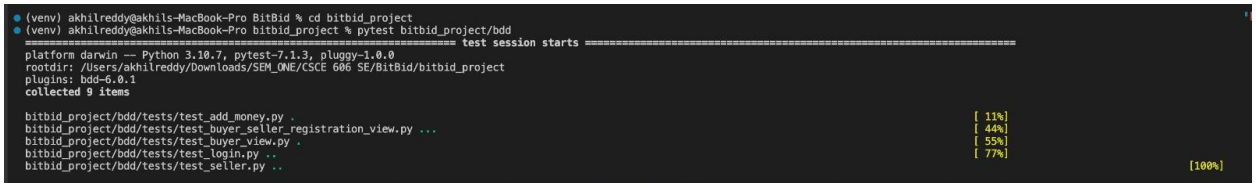
BDD Testing

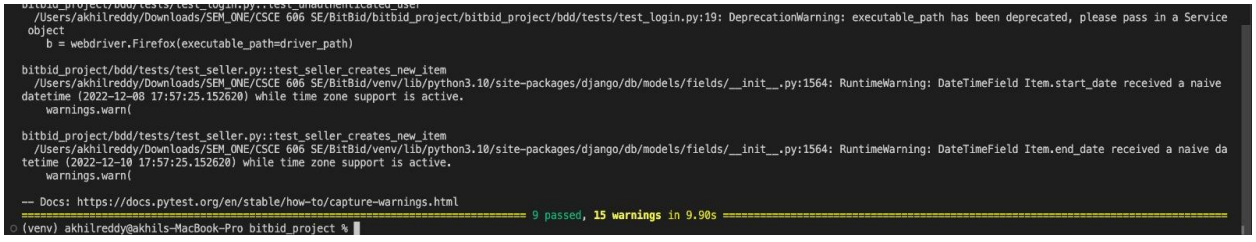
Behavior-driven tests were implemented for all the features. As we are using python for our project, we had to explore what modules to use to implement the BDD tests. We finally chose python's **pytest-bdd** to implement our tests as this module fits our requirements the best.

We have created 6 feature files in total. These feature files cover scenarios like the login feature, buyer and seller registration feature, the ability for a user to add money to his/her wallet, the buyer view feature, and the seller view feature. Each feature file contains one or more scenarios and each scenario is clearly defined with 'Given', 'When', and 'Then' to describe the feature behavior and the expectations of what the feature has to accomplish.

We have then coded these behavior-driven tests in separate test files. Each test refers to the feature file and a particular scenario being tested. In some of these tests, we have made use of the selenium python library along with Mozilla web driver for browser-based automated testing. We have mainly used Django's 'Client' library to create REST requests (GET and POST). The client library also allows the creation of an authenticated test user in order to make these REST requests. Many of our tests require creating fake database records to test certain operations. Therefore, after each test completion, the test flushes the database records before the next test's execution starts.

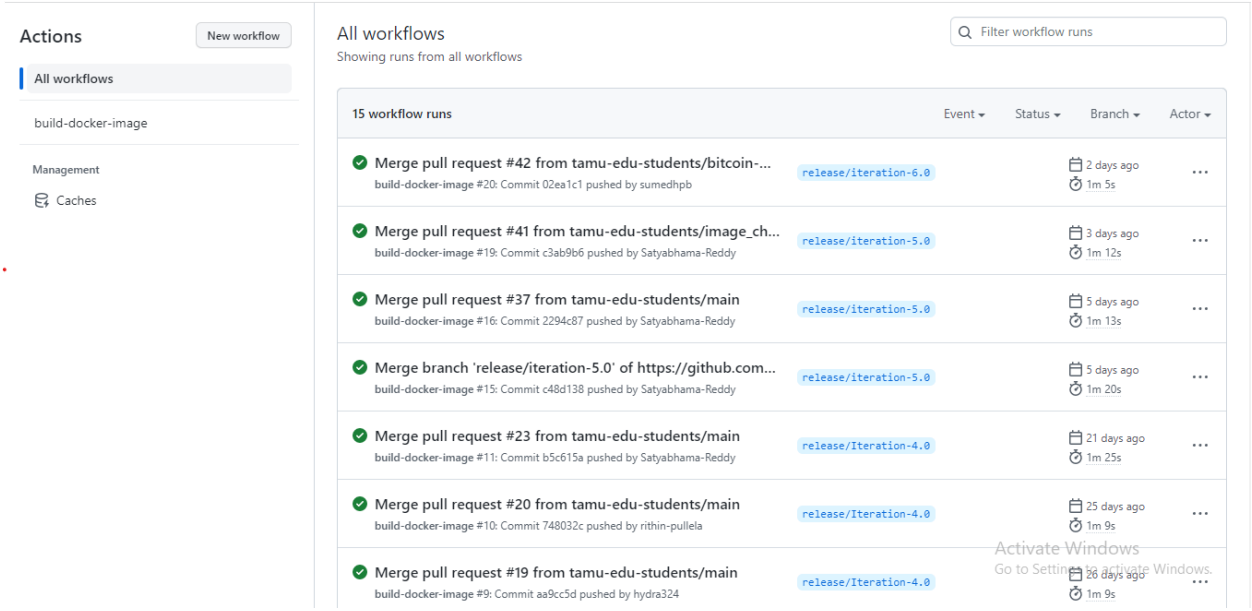
Below is a screenshot of the test cases run:





Configuration Management

The developer generates a new feature branch from master for each new story or task. After completion of the task, the developer commits all changes and pushes them to his/her feature branch. And then after the completion of testing, the developer submits a pull request to the master branch. Following that, the other developers will assess the pull request by either approving or rejecting it. The branch gets merged to the master after the approval. In total there are 23 branches and 6 releases. All release branches start with the prefix release/. Any changes pushed to these branches will be automatically deployed to Heroku as there is a GitHub workflow defined for the same. Below is a screenshot of the builds triggered by each release:



Production Release Process - Heroku

GitHub workflow has been setup for the bitbid repository to deploy to Heroku using HEROKU_API_KEY secret. Every new build is triggered, by pushing the code to /release/** branch which includes docker file and build.yml. A new docker image is built in the workflow and is deployed to Heroku. and can check the status of the build by going into the Actions tab in the git repo.

Tools Used

We used GitHub's Automated Workflow System to deploy our project on Heroku. This automation reduced a lot of redundant work that had to be done for each iteration. Other tools that were used include python's pytest-bdd for BDD Testing, coverage for code coverage report and coinbase-commerce package to enable crypto-payments in our website.

Repository Structure and Deployment

As we are using Python's Django framework, the git repository consists of the following directories and files:

Bitbid_project - It has the project codebase files. The important files and directories present in these are

- manage.py: It consists of the main function. Along with this file this directory also contains other important python files.
- templates: All the HTML files are stored in this directory.
- static: All the CSS files are stored in this directory.
- bdd: All the test files are stored in this directory.

Documentation - It contains all the project documentation and iteration reports.

README -This file briefly describes about the project and the steps to setup the project locally and deploy to Heroku and it also describes the steps to setup test environment.

BitBid Design Doc - It describes the design of our website as described by our client.

Project Links

- GitHub Repository
- Pivotal Tracker
- Bitbid Heroku Deployment
- Presentation & Demo