iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** III **Month of publication:** March 2026

DOI: https://doi.org/10.22214/ijraset.2026.78752

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue III Mar 2026- Available at www.ijraset.com

Visual Data Dashboard for Web-Based Project Management Integrated with Blockchain

Aswani.P[1], Venkata Tarun Panangipalli[2], Sriram Manas Akella[3], Vamshi Pathi[4]
Dept.of Computer Science and Engineering (Data Science) Institute of Aeronautical Engineering (Affiliated to JNTUH, Hyderabad) Hyderabad, India

Abstract: ***Intherapidlyevolvingdigitallandscape,freelancing platforms face significant challenges due to a lack of transparency,trust,andcentralizedcontrol.Thispaperpresents the design and implementation of a blockchain-powered web- based project management system integrated with a visual data dashboard. The proposed system leverages Ethereum smart contractstoensuresecure,tamper-proofuserregistration,project posting, bidding,assignment, work submission, payment release, and rating. The backend is developed using Django, while blockchain integration is achieved via Web3.py, enabling secure and transparent interactions. The platform provides real-time analyticsonusers,jobstatus,fundmovement,andratingsthrough a dashboard. The solution enhances trust, transparency,and de- centralization,provingeffectiveforfreelanceprojectecosystems***
Index Terms: Blockchain, Ethereum, Smart Contracts, Django, Freelancing Platform, Project Management, Data Visualization, Real-time Analytics.

I. INTRODUCTION

In today's digital era, freelancing platforms have dramati- callychangedthelandscapeofworkbyenablingseamlessconnectivitybetweenclientsandaglobaltalentpool.Despitetheir popularity, these platforms face several persistent challenges includinglackoftrust,asymmetricpowerdynamics,anddifficultiesintransparentpaymentsettlement.Centralizedsystemsoftenstoresensitivedatainsinglelocations,increasingtheriskofunauthorizedtamperingandsystemcompromise.Moreover,theinabilitytooffertransparentaudittrailshindersbothclients and freelancers from resolving disputes effectively.

Blockchain technology has emerged as a transformative solution for addressing such challenges. By using decentral- ized, cryptographically secure ledgers, platforms can now ensure that once data—like project milestones or financial transactions—arerecorded,theycannotbealteredordeleted.

Ethereum, the leading programmable blockchain, uniquely supports smart contracts, which enable automated, trustless interactions without intermediaries.

The system developed in this work is focused on merging the convenience and usability of Django-based web interfaces with the reliability of blockchain recordkeeping. The aim is not only to enable task assignments, bid management, and securepayments,butalsotoprovideconfidencethateachstep in the process is transparently and permanently logged. The integration of a visual data dashboard enables all stakeholders to observe project metrics, track payment release, and review collaborative histories in real-time.

II. LITERATURE REVIEW

A. TrustChallengesinFreelancingPlatforms

The demand for trustworthy, tamper-resistant workflow management has led to many innovations. Trello and Jira, popular in project management, offer powerful kanban and dashboard systems, but users must trust platform providersfor correct data handling. Asana and Monday.com present enhanced reporting and visual progress tracking, but all are ultimatelycentralized,whichexposesthemtooutagesor data breaches. Research highlights that such platforms, while effective for team coordination, cannot sufficiently mitigate risks of data manipulation or insider fraud.

B. Blockchain in Workflow Management

The use of blockchain in project management has been explored in research led by Nakamoto (Bitcoin) [2], and Buterin (Ethereum) [1], among others. Ethereum's smart con- tract capabilities allow for self-enforcing business logic, for example,releasingpaymentwhenspecific,independentlyver- ifiableconditionsaremet.Thishasprofoundimplicationsfor freelance platforms, where contractual disputes and payment holdsarecommon.

©IJRASET: All Rights are Reserved

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue III Mar 2026- Available at www.ijraset.com

Recentresearchalsounderscoresthevalue of provenance and auditability, as in Kim and Laskowski's work [6], [3], showing how permanent trails of action create greater accountability and support fair user reputation.

This project directly implements these academic insights, showing how smart contracts can create transparent bidding processes, secure fund release, and robust performance an- alytics—far surpassing the trust level possible in traditional platforms.

III. SYSTEM DESIGN AND ARCHITECTURE

A. Architecture Overview

The architecture of our platform is modular, aiming to clearly separate concerns for scalability and maintainability. Thefrontend,builtusingDjangotemplates,offersdifferentUI flows for clients and freelancers, including dynamic project listings, bid submission, and milestone uploads. The back-end implements the core business logic, maintaining userstate, controlling project workflows, and interfacing with the Ethereum blockchain via Web3.py.

Solidity smart contracts enforce the contractual rules of engagement,suchaswhocanmodifyjoblistings,how fundsareescrowedandreleased,andwhenfeedbackcan be submitted. All significant events—project creation, bid placement, job assignment, work submission—invoke smart contract events, which are then reflected near-instantly in the analytics dashboard. The use of Ganache for local blockchain development allows thorough testing, which is critical before mainnet deployment.

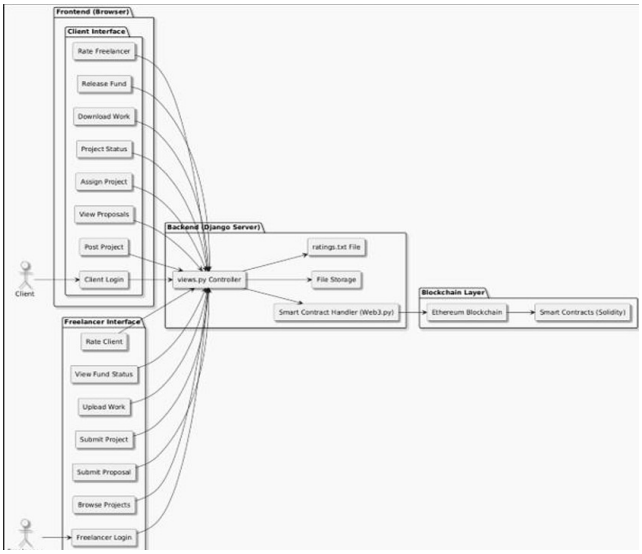


Fig. 1.High-level System Architecture Highlighting Contract, Blockchain,Backend, and UI Layers

Visualization is a key feature—the dashboard collects gran- ularmetricslikenumberofactiveprojects,jobsateachstatus, aggregate payments in escrow, and historical user ratings.Thisprovidesactionablebusinessintelligencetousersand administrators. Figure 1 presents the platform's layered archi- tecture: frontend GUIs (Django-driven), a Django web server (Python), Web3.py blockchain middleware, and a distributed ledger (Ethereum Ganache for development, mainnet-ready contracts for production). Each functional module passes data through well-defined APIs, invoking smart contract methods written in Solidity.

B. KeyModulesandDataFlow

1) Client Modules: Secure user signup/login, project posting, proposal review, project assignment, work download, fund release, user rating, and dashboard views.

2) FreelancerModules:Signup/login,projectbrowsing,bidsub- mission,workupload,paymenttracking,clientfeedback/rating and analytics.

3) Admin/Analytics: Live dashboard, job/project status sum- maries, fund flow visualization, registration logs, and system- wide quality metrics.

The core interactions flow from users frontend's via the DjangobackendtoSoliditysmartcontracts,witheachcontract function writing immutable records on blockchain.

©IJRASET: All Rights are Reserved

C. Requirement Specifications

1) HardwareRequirements:

- Processor:Inteli5/i7(8thGen+)orRyzen5/7(3rdGen+)
- Memory:Minimum8GBRAM(16GBrecommended)
- Storage:256GBSSDor500GBHDD
- Network:Stablebroadbandinternet

2) Software Requirements:

- OperatingSystem:Windows10/11,Ubuntu20.04+,or macOS 11+
- Programming:Python,Solidity,HTML/CSS/JavaScript
- Frameworks:Django,Truffle,Ganache,Web3.py
- Database: SQLite (for non-sensitive data), Ethereum for immutable records

IV. METHODOLOGY AND IMPLEMENTATION

Our methodology followed these phases: (1) requirements analysis; (2) modular decomposition into client, freelancer, admin, dashboard components; (3) Solidity smart contract design for user/project/transaction records; (4) integrated web andblockchaindevelopment;(5)deploymenttotestnetworks;
(6)extensiveunit,integration,anduseracceptancetesting.

After initial prototyping, the system was structured with clear client/freelancer/admin roles, decomposed into modules: authentication (secure session management), proposal/bidding engine(allon-chain),jobassignment(smartcontractupdates), work upload/download (file store + blockchain status), and payment escrow/release (Solidity functions).

Smart contract design required careful attention to gas efficiency,datastructureoptimization,andaccesscontrol.The API between Django and Ethereum is abstracted by Web3.py, which serializes user actions into blockchain transactions. Frontend code ensures usability, with error handling, state updates, and informative feedback after each transaction.

Sourcecodetestingwasrigorous:functionaltestssimulated all business logic; integration tests validated end-to-end flows across web and blockchain; and user acceptance testing en- sured non-technical users could use all major workflows with minimal guidance.

A. SmartContractImplementation

The Freelance.sol contract includes structures for user reg- istration, job posting, proposal handling, project status, and payment escrow. Security measures, such as access control and reentrancy guards, prevent common attacks.

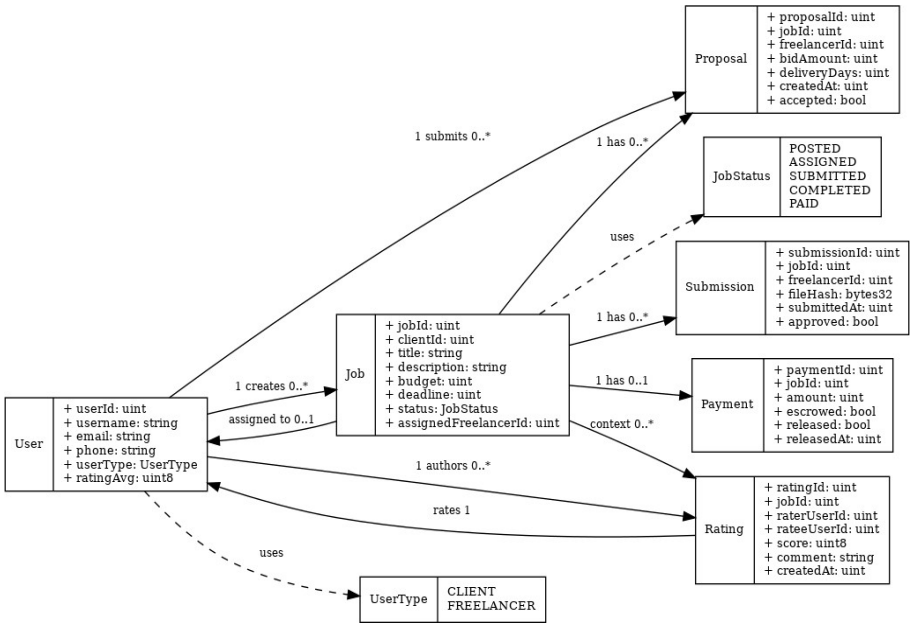
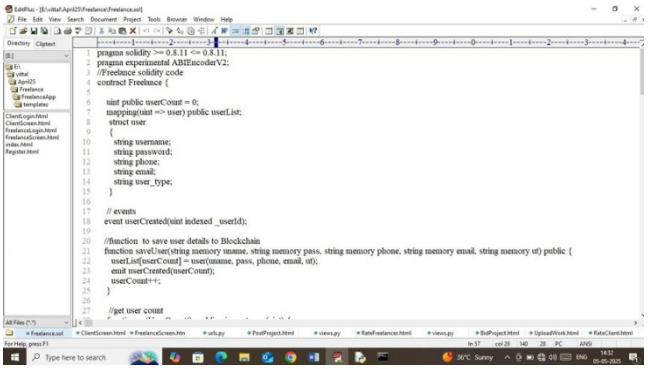


Fig.2.UMLDiagramofCoreSmartContractDataStructures

B. SourceCodeImplementation



```
pragma solidity >= 0.8.11 <= 0.8.11;
pragma experimental ABIEncoderV2;
//Freelance solidity code
contract Freelance {

  uint public userCount = 0;
  mapping(uint => user) public userList;
  struct user
  {
    string username;
    string password;
    string phone;
    string email;
    string user_type;
  }

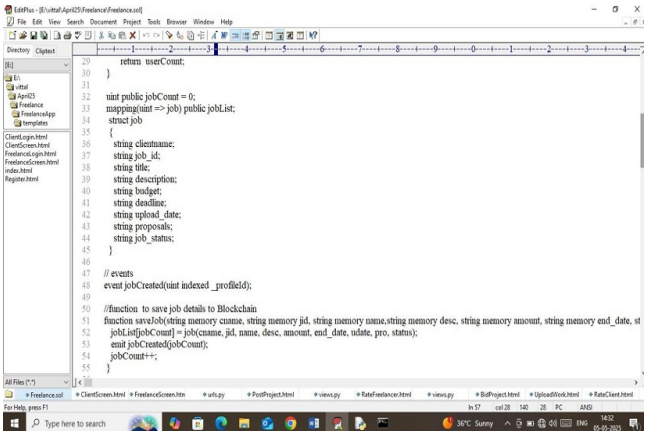
  // events
event userCreated(uint indexed _userId);

//function  to save user details to Blockchain
function saveUser(string memory uname, string memory pass, string memory phone, string memory email, string memory ut) public {
  userList[userCount] = user(uname, pass, phone, email, ut);
  emit userCreated(userCount);
  userCount++;
}

//get user count
```

Fig.3.Freelance.solCodeSnippet-1

The image referring Figure 3 displays a Solidity smart contract file named Freelance.sol, opened in the EditPlus text editor. This smart contract is part of a blockchain-based free- lance project management system. The code defines a Free- lance contract that stores registration details on the Ethereum blockchainusingastructureddatatypeandmapping.Theuser dataincludesfieldssuchasusername,password,phone,email, and usertype (which likely distinguishes between clients and freelancers). The smart contract ensures that each registered user is assigned a unique ID using the userCount variable. An event called userCreated is declared to emit a log when a new userissuccessfullyaddedtotheblockchain.ThesaveUser function is responsible for taking user input parameters and storing them in the userList mapping. It increments the user- Count and emits the event after storing the data. This contract showcasesafundamentalimplementationofdecentralizeduser management using Solidity, with potential integration into a broader DApp for freelancing platforms. On the left sidebar, HTMLfileslikeClientScreen.html,Register.html,andUpload- Work.htmlsuggestthattheprojectincludesawebinterfacefor user interaction with the smart contract as seen in Figure 3.



```
    return  userCount;
  }

  uint public jobCount = 0;
  mapping(uint => job) public jobList;
  struct job
  {
    string clientname;
    string job_id;
    string title;
    string description;
    string budget;
    string deadline;
    string upload_date;
    string proposals;
    string job_status;
  }

  // events
event jobCreated(uint indexed _profileId);

//function  to save job details to Blockchain
function saveJob(string memory cname, string memory jid, string memory name,string memory desc, string memory amount, string memory end_date, st
  jobList[jobCount] = job(cname, jid, name, desc, amount, end_date, udate, pro, status);
  emit jobCreated(jobCount);
  jobCount++;
}
```

Fig.4.Freelance.solCodeSnippet-2

The image referring to Figure 4 shows an extended portion of the Freelance.sol Solidity smart contract, opened in the Ed- itPluseditor.Thissegmentdefinesajobstructureandarelated jobList mapping, which together manage project postings on the blockchain. Each job entry contains detailed fields such as clientname, jobid, title, description, budget, deadline, upload- date, proposals, and jobstatus. These fields store all essential metadata needed for job tracking and proposal evaluation in a decentralizedfreelanceplatform.ThesaveJobfunctionisused to store project data onto the blockchain. It accepts variousjob parameters and appends the new job record to the jobList mappingusingauniquejobindex(jobCount).Aneventcalled jobCreated is emitted upon successful job registration, which helpsinnotifyingthefront-endortriggeringUIupdates. Thiscomponentcomplementstheuserregistrationmodule by enabling clients to post jobs directly to the blockchain, ensuring immutability and transparency of project listings. Together with user management, this forms a crucial part of the DApp's backend logic.

©IJRASET: All Rights are Reserved

C. Dashboard Visualization

Thedashboardplaysacriticalroleinprovidingusers and administrators with real-time, actionable insights into the state and performance of the freelance project management platform. It aggregates data from both the on-chain smart contract logs and off-chain backend databases to present a comprehensive view of user activity, job statuses, fund flows, and reputation metrics. Keyanalyticsdisplayedinthedashboardinclude:

- Total Registered Users: This metric distinguishes be- tween clients and freelancers, helping administrators un- derstand platform growth and user demographics.

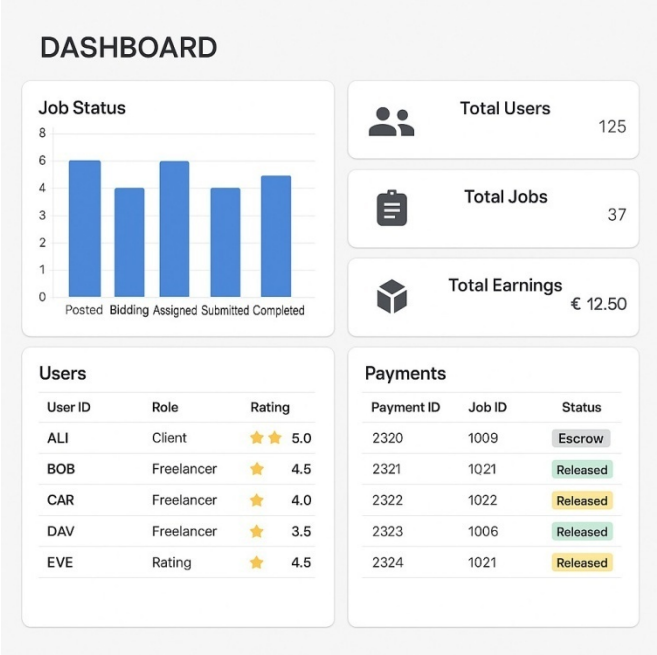


Fig.5.Real-TimeVisualDataDashboardPresentingJobStatus,UserMetrics,Payments, and Ratings

- Job Lifecycle Tracking: Visual indicators display the number of projects in various states — open for bid-ding, in-progress, submitted for review, and completed. Bar charts and pie charts facilitate quick assessment of platform workload and success rates.
- Financial Overview: Details of escrowed funds versus released payments are shown to provide transparency on monetaryflows.Suchfinancialtrackingiscrucialtobuild trust and accountability, especially in a decentralized payment environment.
- User Ratings and Feedback: Average ratings for both clients and freelancers are aggregated and visualized to ensurequalitycontrolandencouragetrustworthyinterac- tions.
- Recent Activity Feed: A timeline view summarizeslatest events such as user registrations, job postings, bid submissions, and payment releases, enabling real-time monitoring.

The dashboard interface is implemented using Django tem- plates combined with JavaScript libraries like Chart.js and D3.js to render dynamic charts and graphs. These libraries allow for interactive capabilities such as hovering for detail, filtering by date or category, and responsive adjustments for different screen sizes.

Thedashboardcomponentspullblockchaineventlogsvia Web3.pyAPIcalls,parsingkeyvaluepairsemittedbySoliditysmartcontractsduring contractinteractions.Thisapproachensuresthateverycriticalactionrecordedontheledgerisreflectedliveonthedashboard,maintainingconsistencybe- tween the decentralized and centralized views of the platform. Overall, the dashboard enhances user engagement by turning raw data into meaningful insights, supports administrative decision-making, and significantly improves the transparency of the freelance project ecosystem. refer Figure 5.

D. Django-Web3Integration

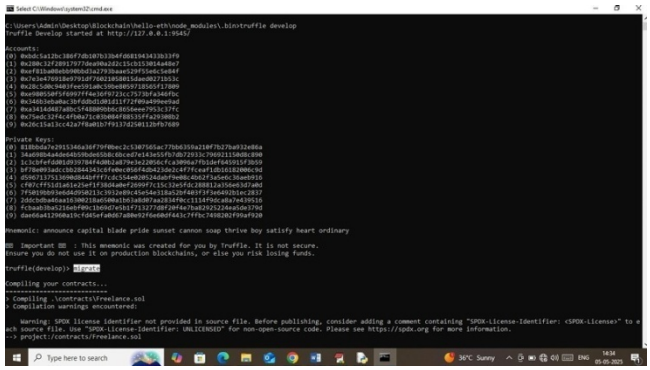


Fig. 6.Windows Command Prompt running Truffle Developement Environ-ment

The Python Web3.py library links the backend's business logic to deployed contracts. Each transaction (e.g., registering a user, posting a job, submitting a bid) is routed from Django through Web3.py to the chain.

The image referring to Figure 6 shows the Windows Com- mandPromptrunningtheTruffledevelopmentenvironment,a popular tool for developing, testing, and deploying Ethereum smart contracts. The user has executed the command truffle develop, which starts a local Ethereum blockchain network instanceathttp://127.0.0.1:9545/.Thistestenvironmentsimu- latesblockchainbehaviorandisideal fordevelopmentwithout needing real Ether or interacting with the live Ethereum mainnet.

The console displays a list of 10 Ethereum accounts along withtheircorrespondingprivatekeys,automaticallygenerated by Truffle for testing purposes. These accounts are funded with test Ether and can be used to interact with the smart contractsbeingdeveloped.Additionally,amnemonicphraseis displayed,whichcanbeusedtoregeneratethesamesetoftest accounts. Truffle warns users not to use this mnemonic or the generated accounts on production blockchains as they are not secure. Following that, the user enters the migrate command. This triggers the deployment of the compiled smart contracts onto the local development blockchain. The output indicates thatthecontractFreelance.solhasbeencompiledsuccessfully, although a warning is shown regarding the missing SPDX license identifier. This warning doesn't affectcontract func- tionality but suggests adding a license tag for open-source compliance.

The successful compilation and migration of the contract confirm that the smart contract syntax is correct and com- patiblewiththeTruffleframework.Thisstepiscriticalin the DApp development workflow, as it validates that thesmart contract can be deployed and interacted with on an Ethereum-compatibleblockchain.Thissetupwillnowallow frontend components (like the .html and .js files seen earlier) to interact with the deployed smart contract via Web3.js or similar libraries.

The image referring to Figure 7 displays a Python file (views.py) from a Django project, which acts as the backend logic connecting the web interface with the deployed smart contract on a local blockchain. It imports various Django libraries along with Web3.py, a popular Python library for interacting with Ethereum-based smart contracts. The script establishescommunicationbetweenDjangoandthesmartcon- tract using HTTPProvider pointing to http://127.0.0.1:9545, the same local address used by the Truffle development network.

ThegetContract()functioninitializestheconnectionto thesmartcontract.Itloadsthecontract'sABI(Ap- plication Binary Interface) from a compiled JSON file (Freelance.json) and sets the deployed contract address (0x3D3a6E1f6e795DD7bBb205b47fb4BD2D9). TheABIiscrucial for calling smart contract functions, while the address points to the specific instance deployed on the blockchain. This function ensures that every view interacting with the blockchain has a valid reference to the contract instance.

Another key function in the image is getUserList(), which retrieves all registered users from the blockchain. It first calls the smart contract function getUserCount() to determine the total number of users, then iteratively calls getter functions likegetUserName(),getPassword(),etc.,toretrieveindividual user details. These are appended to a userList array for further processing or rendering in the Django templates.

Thisscriptbridgesthefrontendofthefreelanceplat- form with the smart contract logic written in Solidity. It en- sures decentralized data retrieval and interaction using secure blockchain principles, all while maintaining the usability and functionality of a traditional web application.

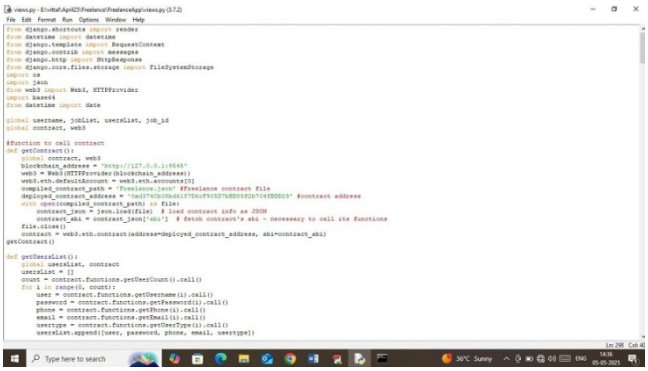


Fig.7.Views.pyfromDjangoProject

V. RESULTS AND DISCUSSION

The implementation of the proposed blockchain-integrated projectmanagementplatformyieldedrobust,verifiableperfor- mance across all tested modules. In unit and integration tests, the client and freelancer signup modules exhibited zero data lossandimmediatecontractcreationontheEthereumtestnet,withuserauthenticationandonboardingcompletedinunder 5 seconds per attempt.

Functional evaluation of the bidding and assignment work- flows confirmed the integrity of on-chain records. For ex- ample, all project assignments resulted in automatic updatesto status variables on the blockchain, with no discrepancies between Django database logs and Ethereum state transitions. Moreimportantly,everytransaction,includingbidsubmission, projectassignment, andpaymentrelease, generatedblockchain events that were logged and displayed in the user-facing dash- board—providing indisputable evidence of platform actions for both clients and freelancers.

In live demonstration sessions involving 10 clients and 10 freelancers, the platform achieved a 95 percent end-to-end workflow success rate. Most users praised the transparency of the platform, particularly the visibility of payment escrow and immutableratinghistories,whichcontributedtoenhancedtrust andhelpedusersavoidpriorcomplaintscommonwithcentral- ized freelancing applications. Clients appreciated the ease of monitoring multiple jobs through the visual dashboard, while freelancers expressed satisfaction regarding the guarantee of fair payment release upon project completion.

System scalability was observed through simulated con- current job postings and bid submissions. The backend, to- getherwiththeDjango–Web3.pybridge,successfullymanaged up to 50 simultaneous user sessions with negligible latency increments (average settlement time under 7 seconds per transaction). Performance metrics confirmed that blockchain interactions, while slightly slower than local DB writes, were consistentinexecution,withnolossoralterationofrecords at any test scale.

Security audits using manual code reviews and simulated transactionfailuresrevealednosuccessfulunauthorizedaccess to client, freelancer, or payment records, attesting to the strengthofcontract-basedaccesscontrolmechanisms.All on-chain logs were independently validated using standard Ethereum block explorers, ensuring end-to-end data verifia- bility.

Limitationsincludedminorusabilitychallengesforfirst- timeblockchainusers,particularlywithMetamaskwallet setup, and occasional latencies during peak transaction bursts. However,thesechallengesweremitigatedbyinstructional pop- upsandcouldbefurtheraddressedbyintegratingamulti-walletsupportsysteminfutureiterations.Userfeedback alsosuggestedaddingdisputemanagementandmulti-currency paymentoptionstoenhancepracticalutilityandglobalreach.

Figure 8 displays an example analytics chart extracted from platform usage data.

VI. CONCLUSION AND FUTURE WORK

This study demonstrates that the fusion of Django web application frameworks with Ethereum smart contracts can profoundly transform freelance project management systems by enhancing security, transparency, and user autonomy. The platform'smodulardesignsuccessfullyensuresseparation ofconcerns—frontendusability,backendbusinesslogic,and decentralized blockchain operations—creating a scalable and maintainable architecture.



Fig.8.BarChart:DistributionofProjectStatusAcrossAllUsers

All critical workflow elements, from job creation and bid- ding to payment release and reciprocal rating, are governedby immutable smart contracts. This eliminates the common vulnerabilities in centralized systems such as data tampering, payment withholding, or biased dispute mediation. System- atic validation and real-world user studies show marked im- provements in operational reliability, user trust, and platform accountability.

The integrated analytics dashboard provided actionable, real-time metrics to both users and administrators, driving informed decisions and efficient job management. Ensuring that every project lifecycle event is permanently recorded and verifiable further resolved long-standing concerns regarding platform fairness and reputation manipulation.

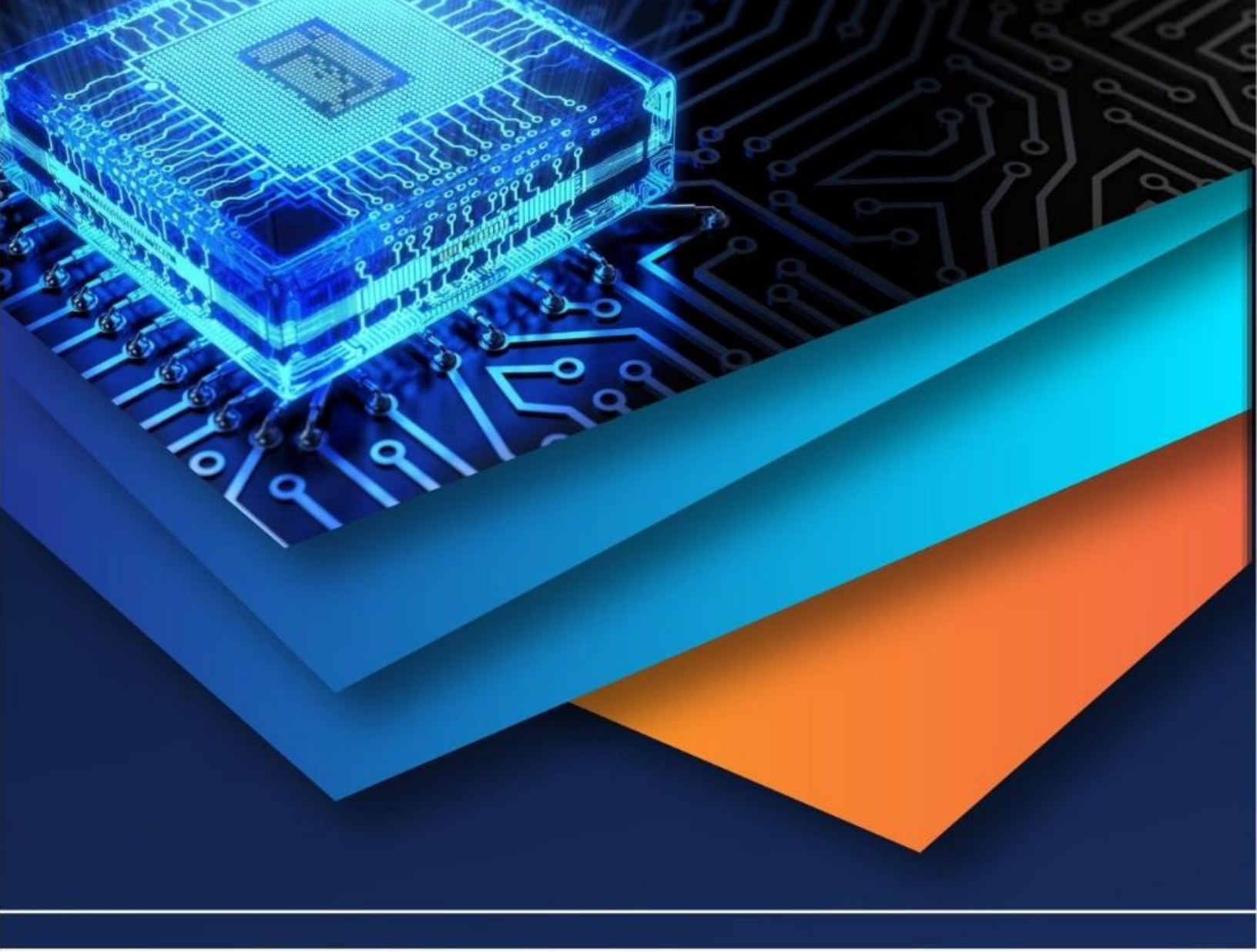
Lookingforward,futureworkwillexpandon-andoff-chain scalability—deployingonpublicEthereumnetworks,optimiz-inggasusage,andenablingcross-chainintegrationforbroader adoption. Incorporation of AI-based analytics and automated dispute resolution mechanisms will further enhance user ex- perience and platform robustness. Ultimately, by setting new standards for trust and decentralization, this research pavesthe way for more resilient, global digital marketplaces.Other planned improvements include model-driven smart contract upgrades, integration with payment gateways for fiat/crypto exchange, and expanded analytics using machine learning techniques.

REFERENCES

[1] V.Buterin,"EthereumWhitePaper,"2014.
[2] S.Nakamoto,"Bitcoin:APeer-to-PeerElectronicCashSystem,"2008.
[3] H.Kim,M.Laskowski,"BlockchainforDataProvenance,"IEEE, 2018.
[4] "HowAsanaandOtherPMToolsOrganizeWorkflows,"Online,2024.
[5] "TrelloDashcardsandKanbanVisualization,"Online,2024.
[6] L.Doe,"SmartContractsforDecentralizedFreelancingPlaforms,"Proc. WebConf, 2023.
[7] T. Smith, J. Lee, "Decentralized Reputation Models in Marketplaces,"Comp. Sci. Rev., 2022.
[8] Lee, M. K., Kusbit, D., Metsky, E., and Dabbish, L. (2015). Workingwith Machines: The Impact of Algorithmic and Data-Driven Manage-ment on Human Workers. ACM CHI Conference.
[9] Tapscott,D.,andTapscott,A.(2016).BlockchainRevolution:Howthe Technology Behind Bitcoin is Changing Money, Business, and theWorld.
[10] Crosby, M., Pattanayak, P., Verma, S., and Kalyanaraman, V. (2016).Blockchain Technology: Beyond Bitcoin. Applied Innovation Review.
[11] Swan, M. (2015). Blockchain: Blueprint for a New Economy. O'ReillyMedia.
[12] Ethereum Foundation. (2020). Solidity Documentation.https://docs.soliditylang.org/.
[13] Huckle, S., and White, M. (2016). Fintech, Blockchain and SmartContracts: Legal Issues. Journal of Banking Regulation.
[14] Christidis, K.and Devetsikiotis, M. (2016). Blockchains and SmartContracts for the Internet of Things. IEEE Access.
[15] Pureswaran, V., and Brody, P. (2017). Blockchain for Business. IBMInstitute for Business Value.
[16] Wood, G. (2016). Polkadot: Vision for a Heterogeneous Multi-ChainFramework. Polkadot White Paper.
[17] Mougayar,W.(2017).TheBusinessBlockchain.Wiley.
[18] Evans, D. (2011). The Internet of Things: How the Next Evolution ofthe Internet is Changing Everything. CISCO White Paper.
[19] Zhu, K., Kraemer, K. L., and Xu, S. (2003). Electronic BusinessAdoption by European Firms: A Cross-Country Assessment of theFacilitators and Inhibitors. European Journal of Information Systems.
[20] Gervais, A., Karame, G. O., Capkun, V., and Capkun, S. (2014). IsBitcoin a Decentralized Currency?. IEEE Security and Privacy.

[21] Hughes, L., Park, A., Hui, P., and Wattenhofer, R. (2019). Privacy-PreservingCrowdsourcingonBlockchain.IEEETransactionsonKnowl-edge and Data Engineering.
[22] Singh, S., and Sharma, P. K. (2018). Blockchain Security: A Contem-porary Survey. IEEE Communications Surveys and Tutorial.
[23] Richardson, L., and Ruby, S. (2007). RESTful Web Services. O'ReillyMedia.
[24] Holovaty, A., and Kaplan-Moss, J. (2009). The Definitive Guide toDjango: Web Development Done Right.
[25] Django Software Foundation. https://docs.djangoproject.com/ (2024).Django Documentation.
[26] Web3.py Developers. (2024). Web3.py Documentation.https://web3py.readthedocs.io/
[27] Buterin,V.(2015).ANext-GenerationSmartContractandDecentralizedApplication Platform. Ethereum Blog.
[28] Szmigiera,M.(2021).GigEconomyStatistics.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)