Plutus Bench

Niels Mündler, OpShin

March 20, 2024

1 Introduction

1.1 Background

In an era where decentralized technologies are rapidly evolving, the development of robust and efficient smart contracts has never been more critical. Plutus Bench thrives to become a solid foundation within the Cardano blockchain ecosystem, aiming to streamline the development process for smart contracts. This report marks the completion of the first milestone in the ambitious journey of Plutus Bench, laying down the foundational stones of project inception and team assembly, alongside extensive market research and community engagement.

1.2 Purpose of this Document

This document serves as a comprehensive report on the first milestone of Plutus Bench. It outlines the project's inception, team assembly, market research, and community feedback, providing a detailed insight into the project's vision and the requirements for the Plutus Bench tool. Furthermore, it establishes a direct line of communication with the community, ensuring that Plutus Bench evolves as a community-centric project.

1.3 Scope

This document is structured as follows:

- **Project Inception:** This section provides an overview of the project's inception, outlining the vision and objectives of Plutus Bench.
- **Team Assembly:** This section introduces the team behind Plutus Bench, highlighting the collective expertise and passion that drives the project forward.
- **Market Research:** This section delves into the market research conducted by the Plutus Bench team, providing insights into the demands and challenges faced by developers working with Cardano smart contracts
- **Planned features:** This section presents the detailed requirements for Plutus Bench, guided by community insights and market analysis, setting a clear path forward for the project.

2 Project Inception

The inception of Plutus Bench is rooted in the vision of streamlining the development process for smart contracts on the Cardano blockchain. The project aims to provide a comprehensive toolset that empowers developers to build, test, and deploy smart contracts with ease and efficiency. At the heart of this vision is the commitment to address the specific challenges and requirements faced by developers working with Cardano smart contracts, ensuring that Plutus Bench is not just a tool but a solution tailored to the needs of the community. The main objectives of Plutus Bench are as follows:

- **Streamline Benchmarking:** Plutus Bench seeks to streamline the process of benchmarking smart contracts, providing tools and utilities that automate the

benchmarking process and provide comprehensive insights into the performance of smart contracts.

- **Community-Centric Approach:** The project is committed to a community centric approach, ensuring that the tools and utilities provided by Plutus Bench are tailored to the specic demands and challenges faced by developers working with Cardano smart contracts.

3 Team Assembly

Due to the widespread usage of the Python programming language, we are able to source developers from outside the community to work on the project. This is a significant advantage, as it allows us to tap into a larger pool of talent and expertise, ensuring that Plutus Bench is developed with the highest standards of quality and efficiency.

The successful development and deployment of a Python-based application interfacing with the Cardano blockchain requires a carefully assembled team of skilled professionals. Given the project's technical complexity and the need for both blockchain-specific knowledge and software engineering expertise, assembling a multidisciplinary team is imperative. This section outlines the key roles within the team, their responsibilities, and the expertise they bring to the project.

Project Management and Blockchain Lead: Niels Mündler

Niels has over 4 years of experience in project management within the tech industry, specializing in blockchain projects. With a strong background in Agile methodologies, Niels has successfully led multiple cross-functional teams, delivering projects on time and within budget while maintaining high-quality standards. He is well known for having delivered several closed and open source projects in the past.

Software Development Lead (Python): Jordan Liu

Jordan is a seasoned software engineer with 12 years of experience, 8 of which are specifically in Python development. He has led multiple software development projects, emphasizing clean code, scalability, and robust architecture. Jordan is passionate about applying blockchain technology to solve real-world problems.

Software Developer: Tomás Herrera

With a Bachelor's degree in Computer Science and 2 years of experience in python development, Tomás specializes in security and efficiency. He is an expert in building scalable solutions.

Smart Contract Expert: Elena Vasquez

Elena is a software developer with 2 years of experience in smart contract development, particularly on the Cardano platform. She has a strong background in Plutus and Haskell, contributing to innovative DeFi projects.

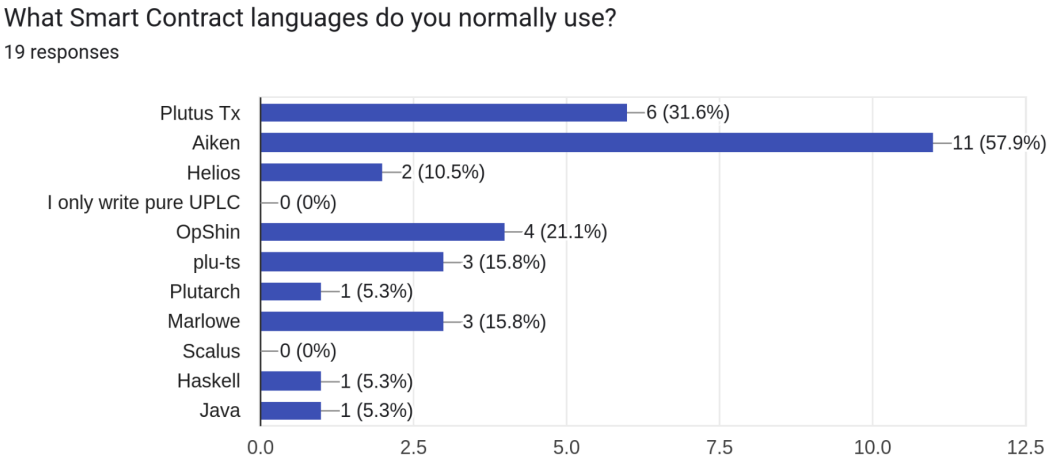
Auditing Partner: Jimmy Koppel

Jimmy completed his Ph.D. at MIT and founded a company focusing on Rigorous Software Engineering and clean code. With his expertise, he counsels the project and provides a sizable network for auditing and quality assurance companies.

As a central point of contact, Niels is responsible for steering the project towards its objectives, ensuring that the team remains focused and motivated throughout the development process. He can be reached through the specifically created email address plutus-bench@opshin.dev.

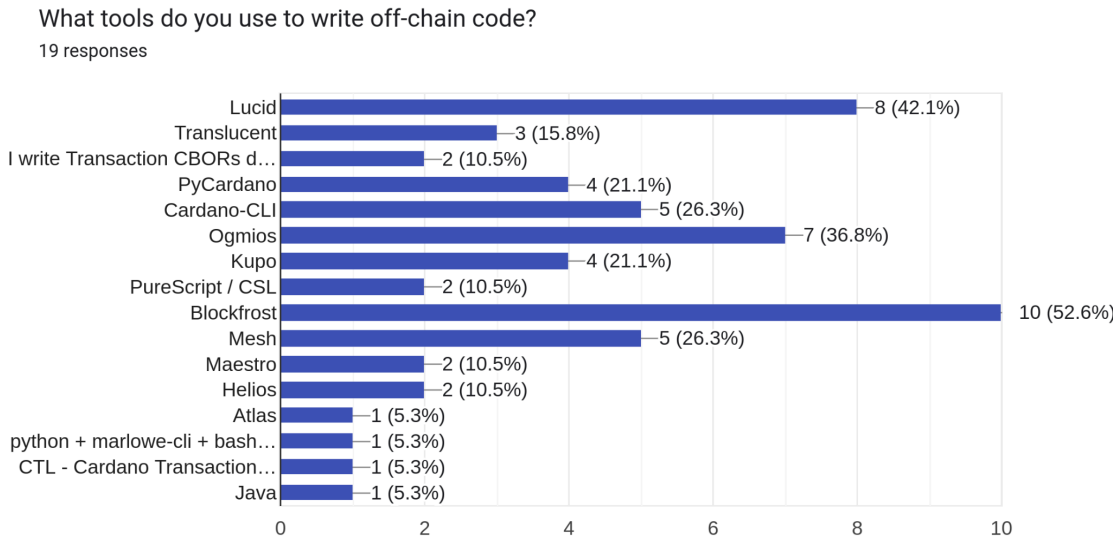
4 Market Research

To assess the current needs of the community, a comprehensive market research was conducted by the Plutus Bench team. The research aimed to identify the specific challenges and requirements faced by developers working with Cardano smart contracts, providing insights that would shape the vision and requirements of Plutus Bench. The research was conducted through a combination of a survey on a variety of development forums, and direct engagement with the Cardano community, ensuring that the insights were drawn from a diverse range of perspectives and experiences. The raw results of the survey are published on the Plutus Bench GitHub repository[1]. The key insights drawn from the market research are as follows:

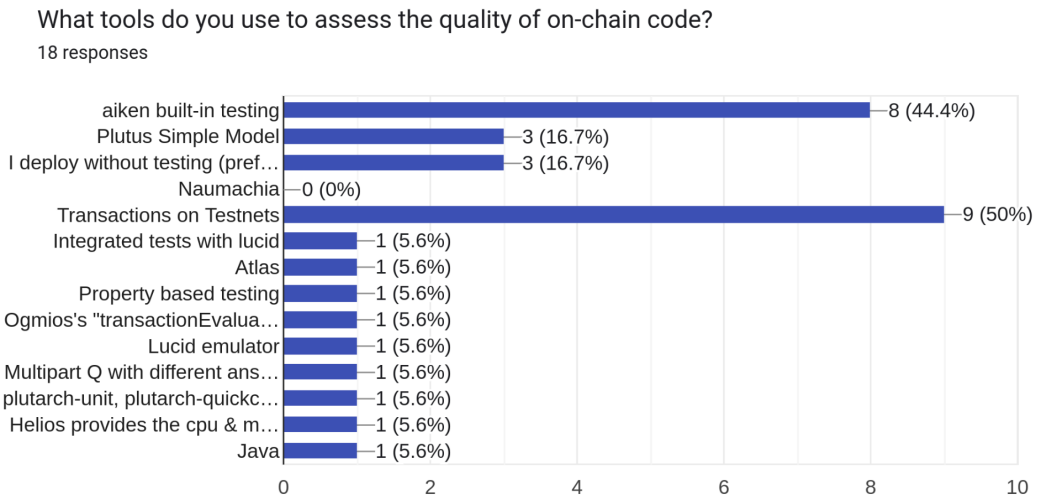


Language diversity: Developers use a variety of languages that compile to Plutus Core, including Haskell, Python, and JavaScript. Although aiken emerges as the most popular language with 60% of users claiming to use it regularly, there is a clear demand for language-agnostic tools that support smart contracts written in any language.

[1] https://github.com/OpShin/plutus-bench/tree/master/report/survey_results.csv

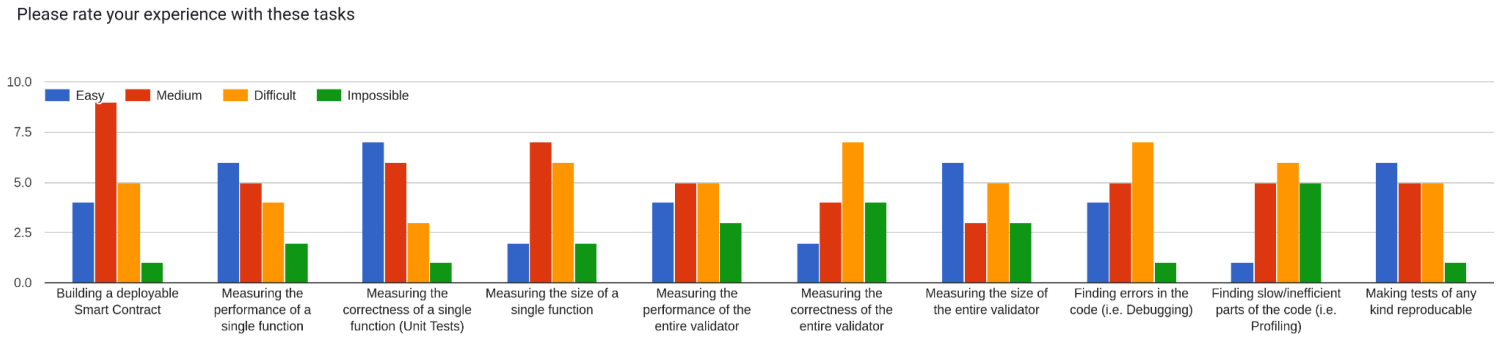


Reliance on Blockfrost API: For building off-chain components, developers rely heavily on the Blockfrost API. This indicates a need for tools that can emulate the Blockfrost API locally, providing a seamless integration of existing off-chain components with Plutus Bench.



Lack of dedicated testing for Smart Contracts: The survey revealed that developers face challenges in testing smart contracts. While the newly developed built-in unit testing in aiken is used by 44% of survey participants, 50% rely on manually submitting transactions to testnets and the remaining answers describe many custom solutions. This indicates a clear demand for a dedicated testing framework that streamlines the process of testing

smart contracts. Note that the aiken testing framework does not yet support testing of full smart contract transactions, only functions thereof.



1. **Difficulty in testing smart contract correctness**: The survey revealed that developers face challenges in testing the correctness of smart contracts. Most describe performing assessments of the correctness of a validator "Difficult" with 4 participants even describing it "Impossible". Similarly, assessing smart contract performance is widely seen as "Medium" to "Difficult" with 3 participants describing it "Impossible". This indicates a clear demand for tools that automate the process of testing the correctness and performance of smart contracts.
2. **Difficulty of line-level assessments:** Most developers describe the process of debugging and profiling smart contracts "Medium" to "Difficult". This indicates a clear demand for tools that provide line-level insights into the smart contracts, streamlining the process of debugging and profiling.

5 Implemented Features

The insights drawn from the market research and community feedback have shaped the vision and requirements of Plutus Bench, ensuring that the project is tailored to the specific demands and challenges faced by developers working with Cardano smart contracts. The features of Plutus Bench are as follows:

- **Dedicated Transaction Testing**: Plutus Bench provides a dedicated testing framework that streamlines the process of testing smart contracts transactions, ensuring that developers can test smart contracts with ease and efficiency. In particular, we provide a simple endpoint that locally evaluates constructed transactions involving smart contracts and returns the result or error logs and performance measurements.
- **Blockfrost API Emulation:** Plutus Bench provides tools that emulate the Blockfrost API locally, ensuring a seamless integration of existing off-chain components with the Plutus Bench toolset. Note that this implies agnostic support for any off-chain component.
- **Language Agnostic Design:** Plutus Bench is designed to be language agnostic, ensuring that developers can build, test, and measure smart contracts written in any language that compiles to Plutus Core.

6 Design Decisions

In this section we detail a few of the design decisions in the Proof Of Concept implementation of Plutus Bench. In general, we have split the implementation into several levels:

- Mock Chain State: This object stores the current UTxO of a faked BlockChain. It provides access to the state with similar methods as the BlockFrostAPI object in the Python blockfrost package. Moreover, it exposes some simple functionality to manipulate the UTxO and the current state of the chain such as the slot number.
- MockFrost Server: The server exposes several REST endpoints that emulate exactly the behaviour of the official BlockFrost REST API. It internally stores several Mock Chain States which can be created and manipulated through extra endpoints.
- MockFrost Client: This client connects to a MockFrost Server and features the creation of fake BlockFrost Clients and manipulation of the Chain State. Since most of the chain-specific functionality emulates BlockFrost, these clients do not need to be re-implemented. Manipulation of the Mock Chain State is as simple as sending a single request to the MockFrost Server.

A detailed overview of the interaction between the different components can be seen in the graphic on the following page. In particular, it can be seen that we allow any programming language to interface with MockFrost through the well known REST protocol. The only requirement to support a new language and framework is to implement the MockFrost Client in the respective target language.

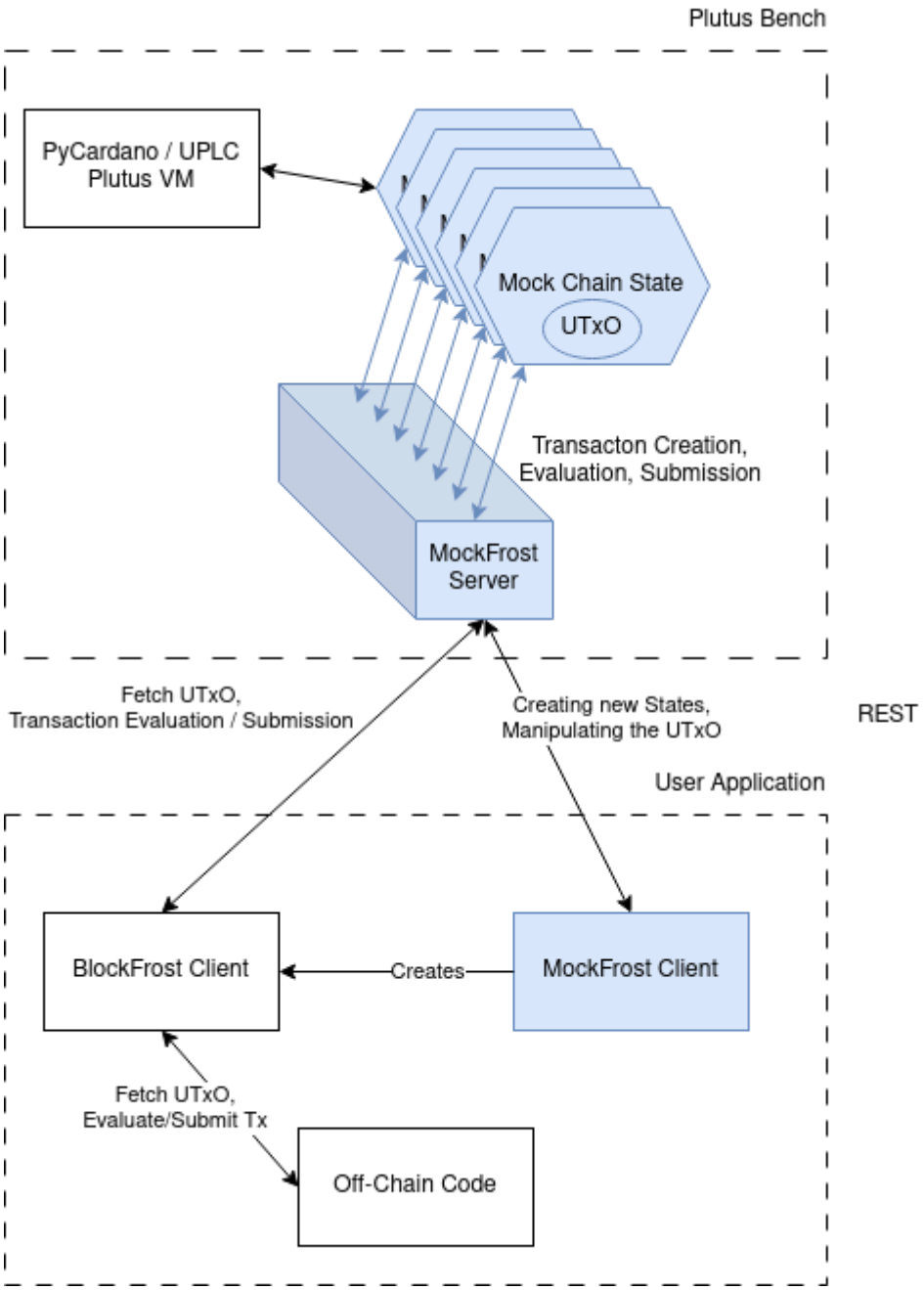
7 Future Expansions for Plutus-Bench

The Plutus-Bench toolset is already useful for a large part of the smart contract development. We outline below several directions in which Plutus-Bench did or could expand given respective interest from the developer community.

- **Variable Plutus Variants:** Many legacy and still relevant Smart Contracts on the Blockchain are written in legacy PlutusV1. Also tools like OpShin are planning to support PlutusV3 soon, with other languages like Aiken already providing support. We therefore integrated PlutusV1 in Plutus-Bench and are planning to also integrate PlutusV3, to enable broader applicability of the tool.
- **Additional Property Based Tests:** We plan to extend the test suite of Plutus-Bench with support for property based tests, which try out a large variety of parameters and can thus become infeasibly slow for practical application. The support can be implemented by a combination of reduced search spaces and increased caching on the side of Plutus-Bench.
- **Line-Level Insights:** Plutus Bench will provide tools that enable line-level insights into the smart contracts, streamlining the process of debugging and profiling. The prototype will execute source Python code in a debugger upon smart contract execution, providing line-level insights into the smart contracts and the ability to debug the code by stepping through lines.
- **Requests from the Community:** We believe the best way to direct further development is to listen to the community desires. We remain committed to listen to the developer community and conduct polls or handle issues submitted to the codebase, as well as staying in discourse on forums such as discord.

8 Conclusion

The completion of the first milestone marks the inception of Plutus Bench, laying down the foundational stones of project inception and team assembly, alongside extensive market research and community engagement. The insights drawn from the market research and community feedback have shaped the vision and requirements of Plutus Bench, ensuring that the project is tailored to the specific demands and challenges faced by developers working with Cardano smart contracts. The features of Plutus Bench are designed to streamline the development process for smart contracts on the Cardano blockchain, providing a comprehensive toolset that empowers developers to build, test, and deploy smart contracts with ease and efficiency.



The abstracted PlutusBench / MockFrost infrastructure. Only Clients need to be implemented in the target language that support manipulating the the UTxO and state which is not part of the standard BlockFrost API. Apart from that, existing Off-Chain code and BlockFrost Clients can be re-used.