Project Development Plan for
Systematic Benchmarking Test Case Generation

Team 1

Emma Willson

Esha Pisharody

Grace Croome

Marie Hollington

Proyetei Akanda

Zainab Abdulsada

October 25th, 2024

Table of Contents

1. Versions, Roles and Contributions

1.1. Version History

Version #	Authors	Description	Date
0	All	Created first draft of document.	October 25, 2024
1	All	First major revision	April 4, 2025

1.2. Table of Contributions

Group Member	Contributions (Sections)
Emma Willson	3
Esha Pisharody	1, 2, 3, 8
Grace Croome	2, 3, 5, 7
Marie Hollington	2, 3, 6
Proyetei Akanda	4
Zainab Abdulsada	2, 3, 8

1.3. Team Information

Group Members	Contact & ID	Roles
Emma Willson	willsone@mcmaster.ca 400309856	Research Lead Developer
Esha Pisharody	pisharoe@mcmaster.ca 400325118	Developer
Grace Croome	croomeg@mcmaster.ca 400313932	Developer
Marie Hollington	hollim3@mcmaster.ca 400320562	Developer
Proyetei Akanda	akandap@mcmaster.ca 400327972	User Interface Lead Developer
Zainab Abdulsada	abduslaz@mcmaster.ca 400313736	Project Manager Developer

2. Team Meeting and Communication Plan

Our team's main communication platforms will be **Teams** and **Discord**. On Discord, we have our Capstone Channel with all team members and supervisors (Dr. Carette and research team). This channel will be used for consistent feedback on project progress including deliverables submitted to Dr. Moradi. Communication through Teams on the other hand will be team-exclusive, for private team-centric updates. For sharing documentation, we will be using Microsoft SharePoint through our Teams channel.

Throughout the course of this project, we plan to have weekly meetings on Wednesdays and Fridays from 12:30-2:30. The meetings will be held in person and on **Teams** in case of member inability to join in person. Our Wednesday meetings will be focused on collaborative work to further advance our goals and may consist of multiple smaller group meetings. Friday meetings will serve as an update for the whole team to ensure that we are all on track, acting as our sprint period boundaries. On Fridays, meeting notes will be recorded and shared on Discord.

GitHub issues will be used for project management. Each issue will correspond to functional and non-functional requirements. Issues will contain a description of what needs to be done and who the task is assigned to.

3. Team Member Roles

Functional Requirements	***Primary Developer(s)***	***Secondary Developer(s)***
The system will contain a hard-coded initial set of EPs written in our proof grammar (MHPG). (P0)	All	
The system will be given a base set of EPs of various types (propositional, natural induction/deduction). (P0)	All	
The system, when given a number of operations and a test from the initial set, will increase the size of the given test case by the provided number of operations. (P0)	All	
The system will generate a set of tests of increasing sizes from the initial set. (P0)	All	
Post-test case generation, the system will translate them into the four PALs. (P0)	All*	
The system will compile the test cases in each PAL. (P0)	All*	
The system will record and display the compilation time of each of these test cases in the four PALs. (P0)	Emma	Proyetei
The system will provide documentation on how to extend the test generator with new classes of tests. (P0)	Grace	Esha
The system will support installation on UNIX environments. (P0)	Zainab	Marie
The system will measure the time and memory complexity of each test case as determined by	Esha	Zainab

comparison of the same base case across increasing sizes. (P1)		
The system will visualize the measured data in graphs and tables on multiple webpages. (P1)	Proyetei	Marie
The system will provide a description for each graph/table displayed on the webpages, explaining the data being visualized. (P1)	Proyetei	Marie
The system will be available as a single downloadable package that enables users to install the complete system. (P1)	Marie	Zainab
The system will support installation on MacOS and Windows environments (P2)	Zainab	Proyetei
The system will require users to select which type of test case is generated. (P2)	Emma	Esha
The system will only allow users to select one type of test case per generation request. (P2)	Emma	Zainab
The system will allow users to select how many versions of each test case is generated, with a lower and upper limit. (P2)	Emma	Grace
The system will validate user input for all parameters and provide an error message if the input is invalid. (P2)	Emma	Zainab
The system will record the translated test cases in their respective PAL and store them in scripts. (P2)	Grace	Esha
The system will record the compilation logs of each test proof and store them in a .txt file, for the purpose of supporting development (error capture), and future research. (P2)	Marie	Emma
The system will incorporate Isabelle as one of the PALs available for translation. (P3)	Grace	Emma

* Functional requirements must be completed for each PAL. The following table outlines developers – PAL responsibility breakdown.

PAL – All member breakdown:

Language	***Primary Developer(s)***	***Secondary Developer(s)***
Lean	Grace	Zainab
Idris	Emma	Esha
Agda	Zainab	Esha
Rocq	Marie	Proyetei

Non-Functional Requirements	*Primary Developer(s)*	*Secondary Developer(s)*
Usability and Human	Proyetei	Marie
Performance	Grace	Emma
Operational and Environmental	Zainab	Marie, Proyetei
Security	All	
Legal	Esha	Zainab

4. Workflow Plan

To effectively manage our GitHub repository and ensure smooth collaboration, we will follow a structured workflow to minimize merge conflicts and streamline code integration. Each team member will create a separate branch for their tasks where they will be committing their code to. All commit messages must be detailed and descriptive. Each commit message should include the associated issue name, which directly corresponds to a functional requirement. This ensures that our work remains traceable and that each commit aligns with the overall objectives of the project. The branch name should be formatted as name-task, for example *proyetei-create-Rocq-proof,* providing clear visibility to each developer's responsibilities. To avoid merge conflicts and ensure that each team member is working with the most current version of the code, every member should perform a git pull from the main branch before creating a new branch. This will pull down the latest changes from the main branch and incorporate them into the new branch, keeping the codebase consistent for all contributors.

Once a team member has completed their task, they will first commit their changes to the specific branch they created. After committing their work, they will create a Pull Request (PR) to merge their branch into the main branch. When naming PRs, the title must clearly describe the task or issue being resolved and reference the corresponding issue. The PR will then be assigned to the project manager, who will review the changes and either accept or reject the request based on the quality and accuracy of the work. If accepted, the project manager will merge the branch into the main branch, ensuring the project stays up to date.

5. Proof of Concept Demonstration Plan

For our proof-of-concept demonstration, we plan to show one test case (eg. nested let statements), as written in a condensed version of what our grammar will become. This will then be translated into each of our target PALs. A challenge with this project is maintaining semantic equivalence across PALs, hence starting with a simple case as our proof-of-concept will allow us to demonstrate the feasibility of translation to each PAL. Our focus over the course of the project will then be making sure this property scales effectively.

A potential risk in our project is unclear presentation of data to an audience that is less familiar with the theory of computer science. Hence, we will also include a mock instance of our webpage with examples of the types of graphs we plan to show. This will help us to showcase

the expected results of our project and their utility in determining the efficiency of proof assistants.

6. Technology

This project will use Haskell, Agda, Lean, Rocq, Idris, Python, and HTML/CSS/JavaScript. HTML, CSS, and Javascript will be used for building webpages, while Python will handle measuring the time and memory complexity. Python will also handle generating the graphs and tables using the Matplotlib library to display these results. The new proof grammar, as well as the test case generation and translation methods will be defined in Haskell. Agda, Lean, Rocq, and Idris will be needed to verify the translated test cases and compile them for performance analysis.

The CI will be implemented using GitHub Actions, with workflows to build, deploy, and test the project. Build workflows will set up the necessary deployment and testing environments. Deployment workflows will be responsible for automating the generation, translation, and compilation of test cases in their respective languages, as well as generating the webpages with the results. Test workflows will ensure that the test case generation and translation methods are functioning properly.

VS code will be used as the default coding environment, leveraging its extensions to simplify code editing and improve our understanding of the syntax of the programming languages we are using. For unit testing in Haskell, we will use HUnit, and for any property-based testing we will use QuickCheck. Shell scripts and archive tools similar to Makeself may be required to create a Linux installer for this project.

7. Coding Standard

The set of test cases and translation will follow the Haskell coding standard.

The measurement of complexities and generation of graphs through Python will use the PEP 8 style guide.

8. Project Scheduling