

MICHIGAN STATE

UNIVERSITY

Project Plan Presentation

SmartSat™ Satellite App Store

The Capstone Experience

Team Lockheed Martin Space

Aidan Delfuoco

Matt Heilman

Will Teasley

Valentino Dore

Colin Williams

Department of Computer Science and Engineering

Michigan State University

Fall 2021



*From Students...
...to Professionals*

Functional Specifications

- This project will provide automated black-box hardware testing to ensure SmartSat™ application compatibility with hardware/software versions.
- The existing web application UI will be updated to specify target hardware and software configuration for SmartSat™ applications.
- Unit tests will be uploaded on the web application and test status will be shown.
- (Stretch goal) RESTful API with token authentication for App Store automation and integration.



Design Specifications

- When uploading a new application to the App Store, target hardware/software configurations will be chosen from a check-box menu, with all versions selected by default.
- The web app will also have a new feature to upload Tinker Testbed forms (test classes).
- When a new app is uploaded, tests are automatically run, and the status can be viewed under the app's description.



Screen Mockup: Upload New or Versioned Application Form

The screenshot shows a web browser window displaying the SmartSet application upload form. The browser's address bar shows the URL <https://smartsat.gov>. The page header includes the SmartSet logo and the Lockheed Martin logo. A search bar is located at the top right of the page. The form itself is titled "Upload New or Versioned Application Form" and contains the following fields and options:

- Action:** A dropdown menu with "Click to upload" selected.
- Image:** A small image of a globe with the filename "fileuploadtest.png" and a delete icon.
- Recommendation:** A text field containing "Recommendation: Update a table; prog and app updates".
- Name:** A text field containing "Rapid Recognition" with a green checkmark.
- Category:** A dropdown menu with "Application" selected and a green checkmark.
- Tags:** A dropdown menu with "Machine Learning" selected and a green checkmark.
- Version Number:** A text field containing "1.1".
- Private Project:** A toggle switch that is currently turned off.
- Format:** Two buttons: "Source" (selected) and "Binary".
- Type:** Radio buttons for "Bin", "Payload", and "Module". "Module" is selected.
- OS Compatibility:** A list of operating systems with checkboxes: "RHEL 7", "Ubuntu Bionic", "ZCU102 SmartSet Linux", and "ZCU102 VxWorks". "Ubuntu Bionic", "ZCU102 SmartSet Linux", and "ZCU102 VxWorks" are checked.
- Description:** A text area containing "Using ML to find".
- Source:** A dropdown menu with "Git Repository Link" selected and a text field containing "git url".
- Direct Upload:** A checkbox that is currently unchecked.
- Save:** A blue button at the bottom right of the form.



Screen Mockup: Upload Test Bed Form

The screenshot shows a web browser window with the URL <http://smartgate.gov>. The page title is "Applications" and it contains a form for uploading test bed configurations. The form includes a "VxRail Configuration" section with a text input field and a "Select VxRail Configuration File" button. Below this is a list of operating systems with checkboxes: RHEL 7 (unchecked), Ubuntu Bionic (checked), ZCU102 SmartGate Linux (checked), and ZCU102 VxWorks (checked). The "Description" field contains the text "Using BTL to test". The "Source" section has a "Repository Link" field and a "Direct Upload" button. A "Submit" button is located at the bottom right of the form.

Applications All of the applications you have downloaded or have access to.

VxRail Configuration:

Select VxRail Configuration File

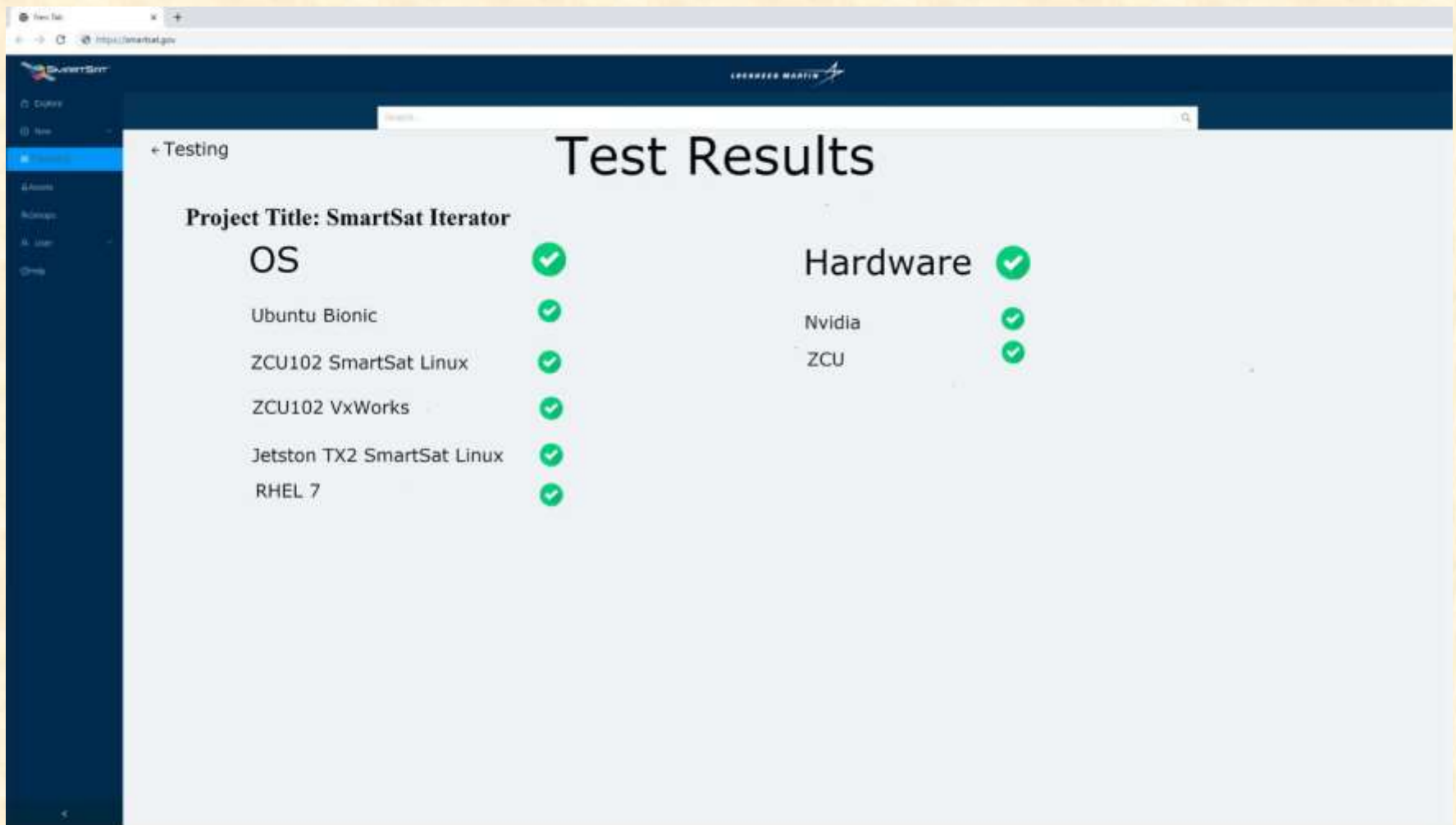
RHEL 7
 Ubuntu Bionic
 ZCU102 SmartGate Linux
 ZCU102 VxWorks

Description:

Source:



Screen Mockup: Executed Test Results Positive



Screen Mockup: Executed Test Results Negative

The screenshot shows a web application interface for 'SmartSat Iterator' project. The page is titled 'Test Results' and displays a list of test environments with their status (pass/fail) and error messages for failed tests.

Project Title: SmartSat Iterator

| Environment | Status | Error Message |
|---------------------------|--------|---|
| OS | ✗ | |
| Ubuntu Bionic | ✗ | Traceback (most recent call last): File "/Users/plasmalazer/Downloads/demo06/-calculator.py", line 156, in <module> parser = build_parser() |
| ZCU102 SmartSat Linux | ✓ | |
| ZCU102 VxWorks | ✓ | |
| Jetson TX2 SmartSat Linux | ✓ | |
| RHEL 7 | ✗ | File "/Users/plasmalazer/Desktop/CSE 450/-proj02-starter/tempCodeRunnerFile.py", line 1 def test_math_nary_empty(self): ^ IndentationError: unexpected indent |
| Hardware | ✗ | |
| Nvidia | ✓ | |
| ZCU | ✗ | Traceback (most recent call last): File "/Users/plasmalazer/Downloads/demo06/-rply/grammar.py", line 1, in <module> from .errors import ParserGeneratorError ImportError: attempted relative import with no known parent package |

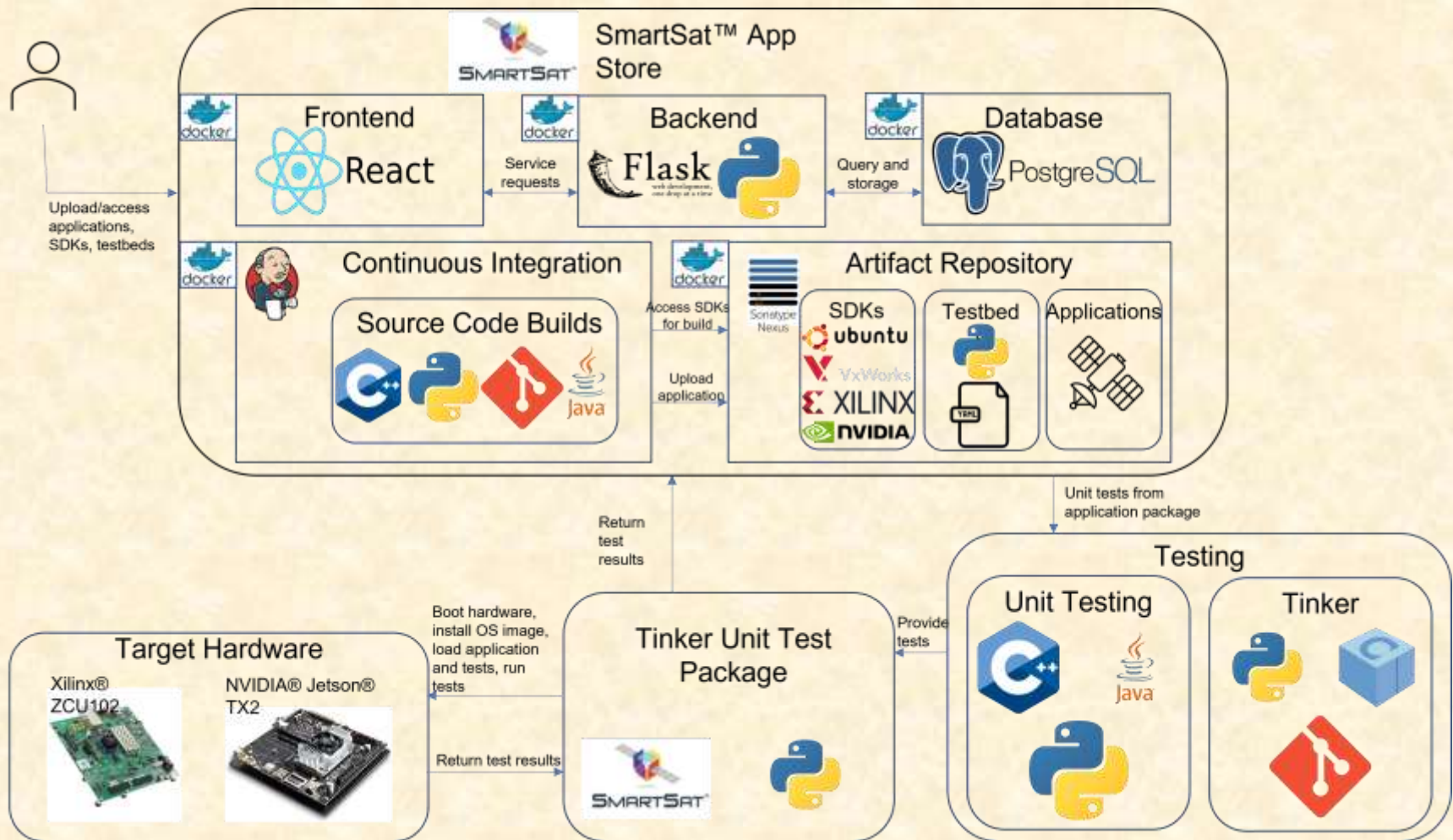


Technical Specifications

- Application developers create tests using Python that interface with the applications executing on the target hardware using built-in SmartSat™ Messaging.
- The SmartSat™ app store will support the creation of new types of testbeds and provide facilities to define instances of the testbed through a YAML configuration file.
- Python package to manage the interactions between the target hardware, unit and Tinker tests, and the App Store.
- The UI changes will be made in React and Flask.



System Architecture



System Components

- Hardware Platforms
 - Xilinx ZCU102
 - NVIDIA Jetson
 - Power Unit to connect hardware in a flight-like configuration
- Software Platforms / Technologies
 - Jenkins
 - Docker
 - PostgreSQL
 - React
 - Flask
 - Conan
 - Tinker



Risks

- Applicable SDKs to test
 - Not all applications are compatible to run with all SDKs on all hardware. How can we keep track of what SDKs an application can run on?
 - We are planning to tell each Tinker testbed which SDKs are compatible. In the UI for app creation, we will ask the user what SDKs/Operating Systems are compatible in a drop-down menu and then we will have to figure out how to configure the Tinker testbed accordingly.
- Building with Tinker
 - Tinker is an Automated Black Box Integration Testing Framework for SmartSat™ Application Packages. It was developed by Lockheed Martin Space internally, so familiarizing ourselves with it will take some time. Before we can begin to incorporate Tinker functionality into the App Store, we need to understand how the tests are written and run.
 - We have received presentation slides outlining the basics of Tinker development. We will have to get familiar with the example test classes developed in Python that incorporate Tinker. The Tinker codebase has been given to us to study and start writing some simple local tests.
- Running Automated Tests
 - Unit tests must be run on all applicable SDK versions whenever an app is updated, as well as on all applicable applications whenever an SDK version is updated. This should be an automated process, and the tests need to all be run on the physical hardware. Figuring out how to push out the unit test to all the hardware serially is a risk because we are unsure how to hook up all the target hardware at one time, and how to store the compatible versions in a configuration file.
 - Using Tinker and ssh to communicate with all the different hardware versions (ZCU102 and NVIDIA Jetson), we can push out the unit tests serially. Once the response comes back from one, we can send the next test. The applicable SDK versions will be specified in a new file that will be stored with the YAML and testbed configuration files.
- Setting all hardware up together
 - We have booted up the ZCU102 and connected to it through the command line interface, but we have not determined the best way to set up the ZCU102 and NVIDIA Jetson at the same time with the power unit.
 - We will need to work with our clients more to figure out the best way to set up all the hardware at the same time.



Questions?

?

?

?

?

?

?

?

?

?

