CZECH TECHNICAL UNIVERSITY IN PRAGUE

FACULTY OF ELECTRICAL ENGINEERING

BACHELOR THESIS

Test Management Tool Prototype for HiL Testing

Togzhan Kuandykova
Electrical Engineering and Computer Science
Supervisor: Ing. Tomáš Haubert, Ph.D.

May 2023

BACHELOR'S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Kuandykova Togzhan**	Personal ID number:	**498074**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Electrical Power Engineering**		
Study program:	**Electrical Engineering and Computer Science**		

II. Bachelor's thesis details

Bachelor's thesis title in English:

Test management tool prototype development for HiL testing

Bachelor's thesis title in Czech:

Prototyp pro HiL testování

Guidelines:

1. Research and comparison of currently available test management tools.
2. Analysis of test management tool needs at Porsche Engineering. Preparation of a prototype concept, aligned with HiL test management needs at Porsche Engineering. Selection of a platform for prototype development.
3. Implementation of the test management prototype for a project at PECZ.

Bibliography / sources:

[1] T. L. T. R. M. W. Andreas Spillner, Software Testing Practice: Test Management, Rocky Nook, 2007.
[2] A. Atar, Hands-on test management with Jira : end-to-end test management with Zephyr, synapseRT, and Jenkins in Jira, Birmingham : Packt Publishing, 2019.
[3] M. Schlager, Hardware-in-the-Loop Simulation: A Scalable, Component-based, Time-triggered Hardware-in-the-loop Simulation Framework, VDM Verlag Dr. Müller, 2008.
[4] A. Axelrod, Complete Guide to Test Automation: Techniques, Practices, and Patterns for Building and Maintaining Effective Software Projects, Apress, 2018.

Name and workplace of bachelor's thesis supervisor:

Ing. Tomáš Haubert, Ph.D. Department of Electric Drives and Traction FEE

Name and workplace of second bachelor's thesis supervisor or consultant:

Date of bachelor's thesis assignment: **14.02.2023** Deadline for bachelor thesis submission: __________

Assignment valid until: **22.09.2024**

Ing. Tomáš Haubert, Ph.D.	doc. Ing. Zdeněk Müller, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
Supervisor's signature	Head of department's signature	Dean's signature

III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce her thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student's signature

Declaration

“I hereby declare that this bachelor thesis is the product of my own independent work and that I have clearly stated all information sources used in the thesis according to Methodological Instruction No. 1/2009 – “On maintaining ethical principles when working on a university final project, CTU in Prague “.

Date: Signature:

Abstract:

The popularity of HiL (Hardware-In-The-Loop) testing grew with the improvements in the automotive field, along with demands for extensive testing of the newly innovated software, cost reduction, and improvement of time efficiency.

This bachelor's thesis will investigate relevant test management practices and commercially available tools. Then, based on the gained insights, a test management tool based on Porsche Engineering project needs is developed.

Table of Contents

List of figures

List of Tables

List of abbreviations:

PECZ	*Porsche Engineering Czech Republic*
HIL	*Hardware-In-the-Loop*
ECU	*Electronic Control Unit*
PCM	*Powertrain Control Module*
R&D	*Research and Development*
ADAS	*Advanced Driver-Assistance Systems*
SQL	*Structured Query Language*
WIP	*Work in Progress*
XP	*Extreme Programming*
JQL	*Jira Query Language*
ADO	*Azure DevOps*
TIS	*Tools Integration System*
API	*Application Programming Interface*
REST	*Representational State Transfer*
CRUD	*Create, Read, Update, Delete*
HTTP	*Hypertext Transfer Protocol*
URL	*Uniform Resource Locator*
HTML	*Hypertext Markup Language*

Chapter 1: HiL Testing and Test Management Theory

1.4 Testing in Automotive Industry

Testing is a vital part of the vehicle development process, it is needed to make sure the product is durable, reliable, is following set regulations, and helps to catch errors at each step of R&D. Traditional testing took several forms, some of them are vehicle performance test, braking capability test and infotainment system tests, that require an actual driver or a crash test with a dummy mannequin.

As the logic of the car software grew in complexity, demand for testing increased exponentially. The development of multiple car systems interconnected with each other required testing of the BUS communication between them. [1] With the innovations in power systems, there arrived a need for protection systems to catch voltage spikes and faults, to minimize the damage caused to the power network. [2]

Traditional testing cannot cover all the test cases that the above-mentioned variables create. Even if they could, it would be inefficient timewise. Outdated testing methods cannot test vehicle components that are in development, while the further the process of vehicle development is, the more expensive are the changes to implement.

Thus, using traditional testing, we test the final product, which will reveal errors not caught beforehand, and will be more expensive for the manufacturer. [3] This is seen on the curve in Figure 1, which shows the relation between the cost of change and the development stage, also known as Boehm's curve.

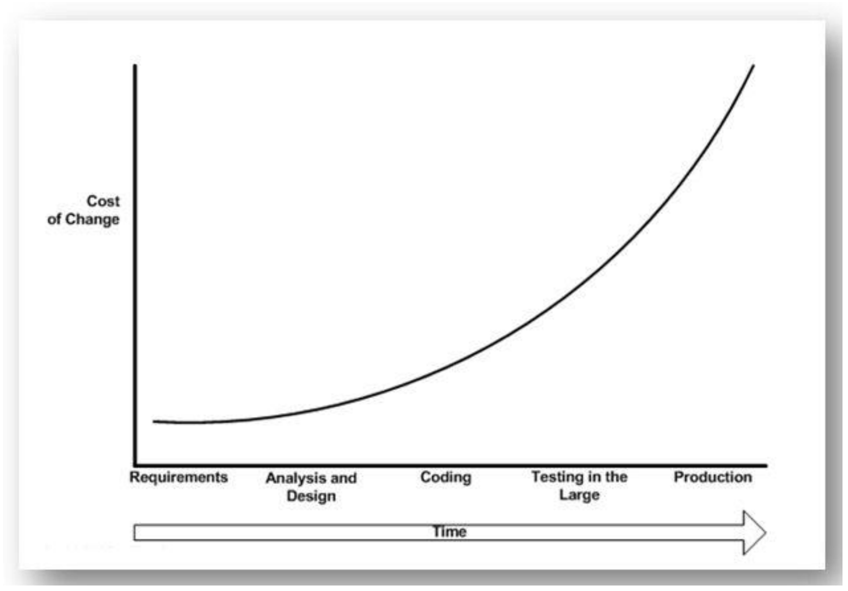


Figure 1. Cost of Change vs Time graph with the Boehm's curve [3]

1.5 Hardware-in-the-loop Testing

Previously in the vehicles, mechanical actuators were used, then the electrification of the vehicles happened, and the gradual change to electromechanical and electrical control units started. ECU is an electronic control unit, that contains some programmed logic in its computer chip, that is responsible for control over vehicles' electrical systems. Based on the price point of the vehicle, the vehicle will contain a different number of ECUs. Low to mid-end vehicles can contain around thirty ECUs, while for a high-end luxury vehicle, the number can go up to around a hundred. [4]

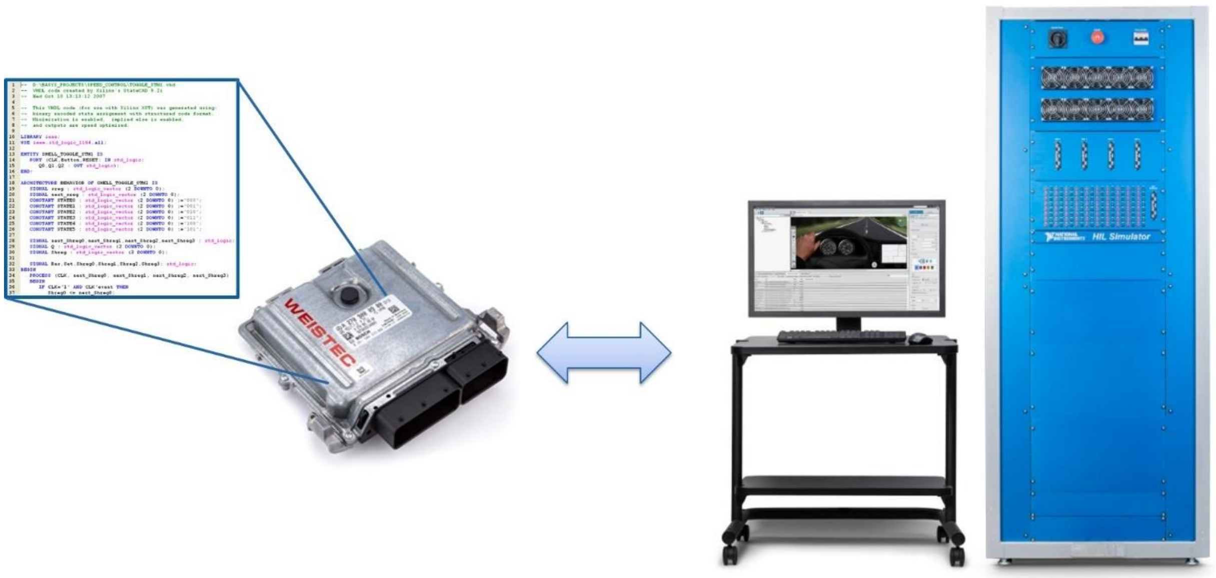
Nowadays, ECUs control the entertainment and comfort systems in a vehicle. Some ECUs can be of the essential type, such as engine control, airbag deployment, power steering, braking control, and antilock braking system. These systems need the most precise testing to ensure the safety.

```
-- D:\BASYS_PROJECTS\SPEED_CONTROL\TOGGLE_STM1.vhd
-- VHDL code created by Xilinx's StateCAD 9.2i
-- Wed Oct 10 13:13:12 2007

-- This VHDL code (for use with Xilinx XST) was generated using:
-- binary encoded state assignment with structured code format.
-- Minimization is enabled,  implied else is enabled,
-- and outputs are speed optimized.

LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY SHELL_TOGGLE_STM1 IS
   PORT (CLK,Button,RESET: IN std_logic;
      Q0,Q1,Q2 : OUT std_logic);
END;

ARCHITECTURE BEHAVIOR OF SHELL_TOGGLE_STM1 IS
   SIGNAL sreg : std_logic_vector (2 DOWNTO 0);
   SIGNAL next_sreg : std_logic_vector (2 DOWNTO 0);
   CONSTANT STATE0 : std_logic_vector (2 DOWNTO 0) :="000";
   CONSTANT STATE1 : std_logic_vector (2 DOWNTO 0) :="001";
   CONSTANT STATE2 : std_logic_vector (2 DOWNTO 0) :="010";
   CONSTANT STATE3 : std_logic_vector (2 DOWNTO 0) :="011";
   CONSTANT STATE4 : std_logic_vector (2 DOWNTO 0) :="100";
   CONSTANT STATE5 : std_logic_vector (2 DOWNTO 0) :="101";

   SIGNAL next_Shreg0,next_Shreg1,next_Shreg2,next_Shreg3 : std_logic;
   SIGNAL Q : std_logic_vector (2 DOWNTO 0);
   SIGNAL Shreg : std_logic_vector (3 DOWNTO 0);

   SIGNAL Res,Set,Shreg0,Shreg1,Shreg2,Shreg3: std_logic;
BEGIN
   PROCESS (CLK, next_Shreg0, next_Shreg1, next_Shreg2, next_Shreg3)
   BEGIN
      IF CLK='1' AND CLK'event THEN
         Shreg0 <= next_Shreg0;
```

Figure 2. Vehicle ECU depicted on the left [5]

Hardware-in-the-loop testing is a way to test the ECU without the actual driver on different stages of development. ECU is connected to a HiL machine, that sends input signals representing different test cases and collects the output. Output properties are then automated and studied, and the result contains data on the failed and passed testcases.

HiL machine can put the ECU through scenarios that are not easily reproducible via traditional testing like earthquakes, weather conditions, different types of roads, and incline. Because the HiL machine can test the ECU at different development stages. According to Boehm's curve the cost of change in case of error is less costly. [3] HiL simulations is less expensive than validation. The setup configuration and the test on the HiL after being set up can run on their own. Figure 3 shows an ECU and the HiL machine that it is connected to for tests.



```
-- D:\BASYS_PROJECTS\SPEED_CONTROL\TOGGLE_STM1.vhd
-- VHDL code created by Xilinx's StateCAD 9.2i
-- Wed Oct 10 13:13:12 2007

-- This VHDL code (for use with Xilinx XST) was generated using:
-- binary encoded state assignment with structured code format.
-- Minimization is enabled,  implied else is enabled,
-- and outputs are speed optimized.

LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY SHELL_TOGGLE_STM1 IS
   PORT (CLK,Button,RESET: IN std_logic;
      Q0,Q1,Q2 : OUT std_logic);
END;

ARCHITECTURE BEHAVIOR OF SHELL_TOGGLE_STM1 IS
   SIGNAL sreg : std_logic_vector (2 DOWNTO 0);
   SIGNAL next_sreg : std_logic_vector (2 DOWNTO 0);
   CONSTANT STATE0 : std_logic_vector (2 DOWNTO 0) :="000";
   CONSTANT STATE1 : std_logic_vector (2 DOWNTO 0) :="001";
   CONSTANT STATE2 : std_logic_vector (2 DOWNTO 0) :="010";
   CONSTANT STATE3 : std_logic_vector (2 DOWNTO 0) :="011";
   CONSTANT STATE4 : std_logic_vector (2 DOWNTO 0) :="100";
   CONSTANT STATE5 : std_logic_vector (2 DOWNTO 0) :="101";

   SIGNAL next_Shreg0,next_Shreg1,next_Shreg2,next_Shreg3 : std_logic;
   SIGNAL Q : std_logic_vector (2 DOWNTO 0);
   SIGNAL Shreg : std_logic_vector (3 DOWNTO 0);

   SIGNAL Res,Set,Shreg0,Shreg1,Shreg2,Shreg3: std_logic;
BEGIN
   PROCESS (CLK, next_Shreg0, next_Shreg1, next_Shreg2, next_Shreg3)
   BEGIN
      IF CLK='1' AND CLK'event THEN
         Shreg0 <= next_Shreg0;
```

Figure 3. Vehicle ECU on the left and a PC connected to the HiL machine on the right [5]

1.6 HiL Test Management

Tests run on the HiLs have different execution and preparation time. In most cases development is done in a parallel way. Several models are developed and tested at the same time. So correct test management can help reduce costs and be more efficient with equipment and outsourcing services.

Test management usually involves activities, such as creating and maintaining component/project cycle information, test execution support such as test execution status capture, time managing the tests, ensuring traceability and visibility of the testing, and reporting the test results. [6]

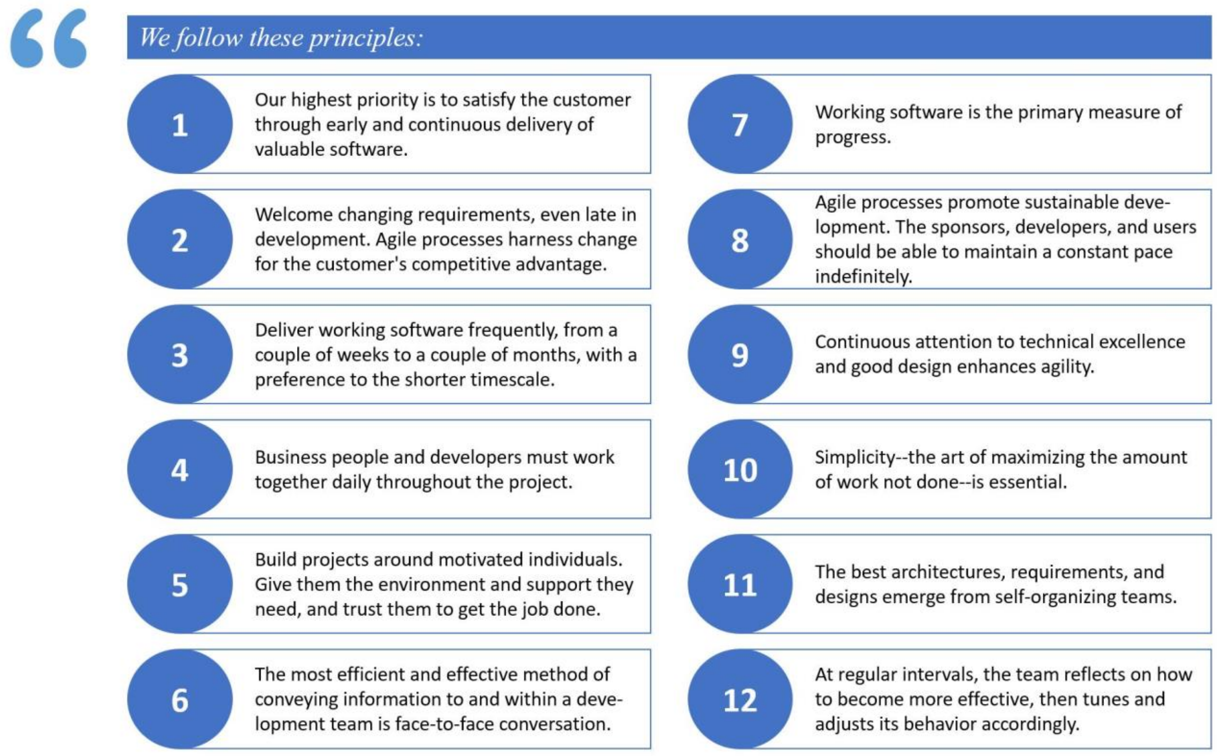
Chapter 2. Test Management Practices and Commercially Available Tools

2.1 Agile Framework in Engineering

Framework in engineering is an approach to development. In other words, a template of sequential actions that helps to efficiently develop products or provide services and meet customer expectations. Porsche Engineering applies Agile framework.

A lot of the modern engineering management frameworks originate in software development processes. According to Williams [7] teams choose a set of agile practices or make a hybrid of them to suit the work that they are assigned instead of sticking to strict guides of one practice.

Agile manifesto principles are described on the Figure below:



We follow these principles:

1 Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

2 Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

3 Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

4 Business people and developers must work together daily throughout the project.

5 Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

6 The most efficient and effective method of conveying information to and within a deve-lopment team is face-to-face conversation.

7 Working software is the primary measure of progress.

8 Agile processes promote sustainable deve-lopment. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

9 Continuous attention to technical excellence and good design enhances agility.

10 Simplicity--the art of maximizing the amount of work not done--is essential.

11 The best architectures, requirements, and designs emerge from self-organizing teams.

12 At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Figure 4. Agile manifesto principles [8]

Description of principles for a successful Agile framework implementation based on Williams [7]:

1. It is an iterative improvement method, where each iteration is followed by a release.
2. Iterations are kept short to compile customer feedback on the latest release and analyze implementation of new needs of the customer or enhancements.

In the book “Agile Software Development” Alistair Cockburn [9] describes that shorter cycle of iteration and project development have been linked to higher success rates as they improved customer satisfaction and feedback, as well as productivity in the working team.

3. Each iteration is like a small project, that includes a full project cycle with elements like analysis of requirements, design, implementation, test, customer acceptance.

Some of the examples of anecdotal data on agile techniques is in books “Agile Estimating and Planning” [10] , “Scrum: The Art of Doing Twice the Work in Half the Time”. [11] These books have case studies on how Agile teams have better predictability, flexibility, productivity, and customer satisfaction.

Empirical findings in this study state that:

- Agile methods are applicable with the same level of success to small and big teams.
- Agile can handle the management and work on reliable and safety-critical projects when the performance requirements are planned out early.
- Agile methods need less formal training and once a handful of team members are agile-proficient it can be taught through mentorship.

Same study states the warning for agile teams:

- Agile cannot survive without the culture where people trust and communicate.
- Low morale expressed during the daily stand up meetings points is one of the signs of poorly integrated Agile methodology in the team.

2.2 Lean Management in Engineering

Lean management – first popularized in Japan, after the Second World War as a new technological spike was crucial for the future of the Japanese economy, the concept was created and implemented by Taiichi Ohno. This management strategy has defined principles and purposes, that serve to get rid of «waste». «Waste» represents steps in manufacturing that do not add value to the product but takes up time and resources. [12] Figure 5 describes main principles of Lean management.

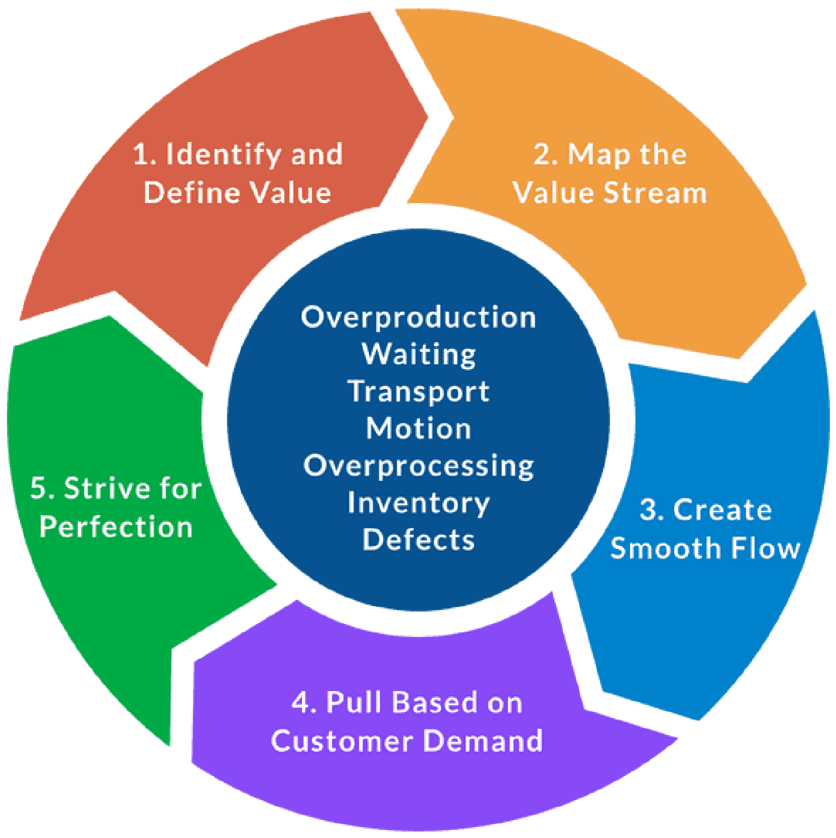


Figure 5. Lean Management principles [13]

2.3 Kanban Board Concept Introduction

Kanban is a framework of a project management layout that can be used with the guidelines of Lean, Agile development or their combination based on the team workflow. Book "Kanban just-in-time at Toyota : management begins at the workplace" [14] states that an engineer at Toyota, Taiichi Ohno, developed Kanban to help the company catch up with the efficiency of its competitors and came up with the concept of a planning tool to create a visualization of the workflow, which would then allow for optimization of the work process. Example of a Kanban board is below, on Figure 6 below.

A Kanban board can be a physical board with sticky notes or a digital table with digital cards. The sticky notes or digital cards are spread based on their stage in the development lifecycle.

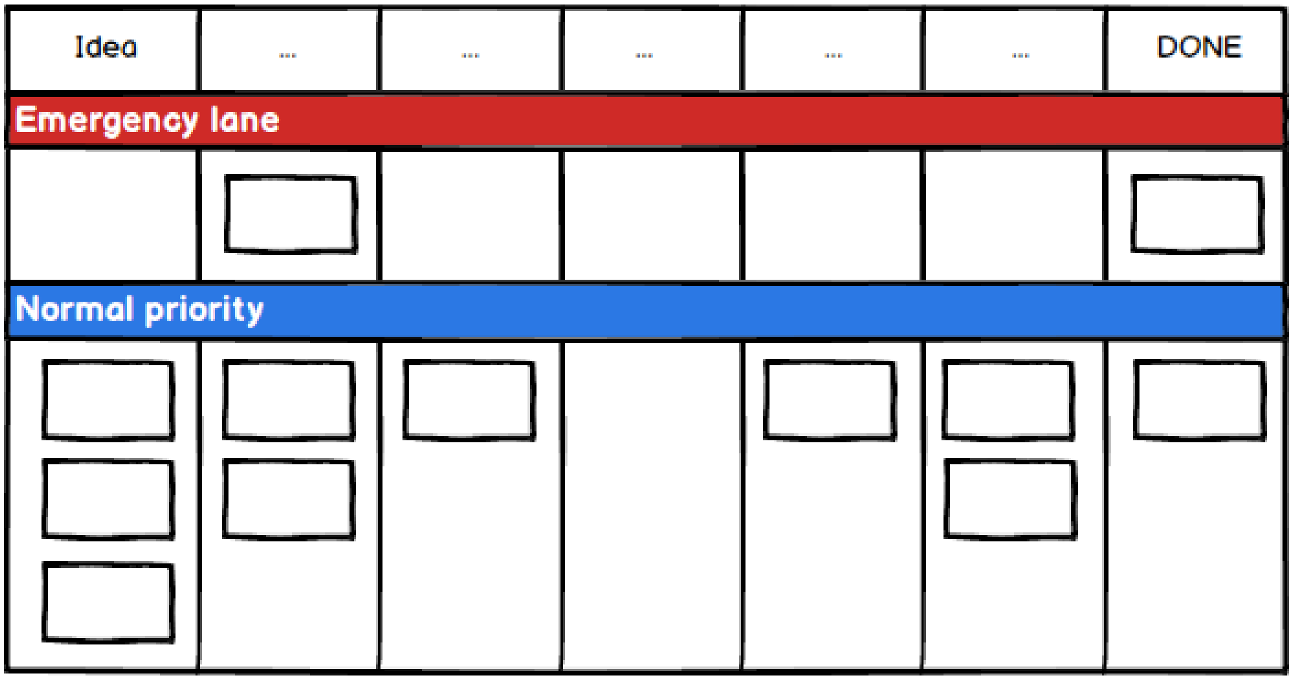


Figure 6. Generic, simplified example of a Kanban board [15]

The table below describes Kanban components

Table 1. Kanban keyword description

Kanban component:	**Usage:**
Cards	Represent user stories, bugs, or tasks. In agile development, a card usually contains a user story that describes customer expectations for a software feature.
Columns	Represent development stages. For example: backlog or queue, in development, in review, done.
WIP (Work in Progress)	A numeric limit that can be applied to the columns that is set up to help control the flow and pace of work and not overwhelm the team to take every task right away. For example, setting WIP of 3 for a column “In Development” would mean that there can only be 3 tasks in development at the same time.
Swimlane	Division of rows in Kanban board that can be used to manage several projects or teams at the same time, also can be used to sort tasks by priority, urgency, releases. On figure 4, swimlanes are for sorting the tasks based on their priority.

2.4 Usage of Kanban in project management.

Anderson [16] states that the approach of using Kanban board is different for Lean and Agile management. Main difference of usage is based on the core values of these frameworks.

In the Lean Kanban described in “Lean from the Trenches: Managing Large-Scale Projects with Kanban” [17] work tasks move through the board linearly from left to right passing all the states of development before being complete.

On the other hand, Agile Kanban is more centered on its principles of having smaller tasks manageable in a shorter time and an iterative feedback loop. In Agile Kanban, cards on the board don't represent the whole task, but a small, manageable in a short time chunk of it. Also, the card can move non-linearly throughout the board. For example, the card could return to the "development" column if it failed the customer review in the "review" column.

2.5 JIRA by Atlassian

JIRA is a subscription software, which was initially created for solving the ticketing and bug tracking management. Over time, JIRA adapted several packages for agile project management, DevOps, software development, and Confluence which is a documentation tool. As of 2020 and 2021, JIRA was rated as the most highly recommended agile tool [18].

JIRA Software package includes:

- Integration with development tools
- Agile/Kanban boards
- Release hub for software versions

Projects in JIRA can be created be team-managed and company-managed. Team-managed projects are for the most part autonomous and the team sets up the projects and workflow. Company-managed projects are for standardizing the confluence (knowledge database) and the setup that is used by several teams.

In JIRA Software, there is an administrator who has all the rights to create and change projects. Other members see the parts of the project that they have access to. JIRA tickets describe a task that needs to be done or an issue to solve. Ticket usually contains an issue summary, issue details, assignee, and custom fields. Custom fields can be the priority of the task, amount of time dedicated for the task, category, release version, or queries to sort tasks based on some component.

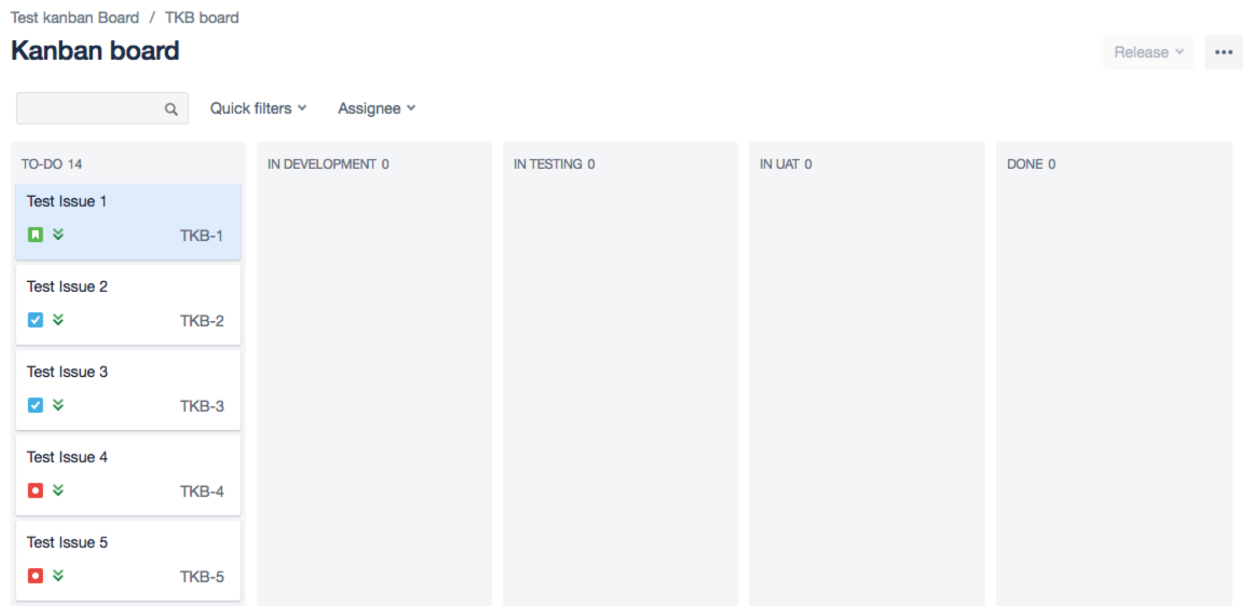


Figure 7. Example of a Kanban board in JIRA [19]

Best practices for JIRA project management are [18]:

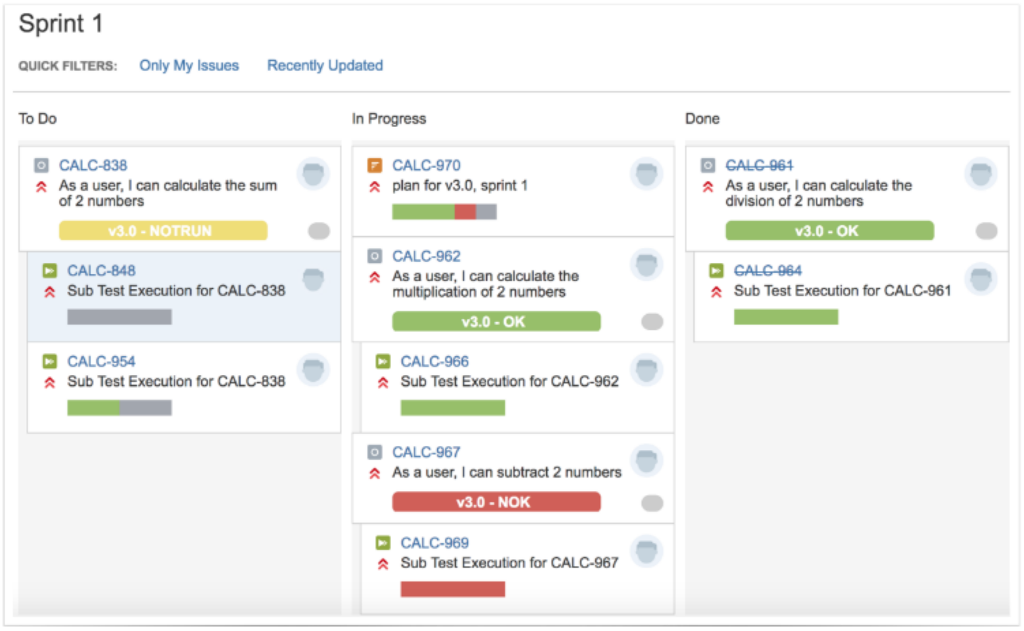
1. Creating issues and linking those issues that have a relationship to each other for ease of planning and communication between developers, teams, and stakeholders.
2. Atlassian documentation states that [20] JIRA has a built-in reporting system which can help visualize the track of development to manage team deadlines and workload. For example, there are reports on the number of created vs resolved issues, time of issue resolution, user workload. Continuous improvement of Agile reports can be set up to optimize team work. All of the reporting possibilities are mentioned in Appendix A.
3. Support of agile methods. JIRA supports such agile methods as Scrum, ScrumBan, XP.
4. Backlog task prioritization. Determining an order that the tasks should be executed in. Taking up on the bigger, more valuable tasks first the team brings more value to the customer. Product backlog is constantly supervised by the development team. Changes are implemented by collaboration of product owner and the development team.

5. It is possible to extend JIRA capabilities and fine-tune it for the team’s usage via external plugins and JIRA add-ons. It is also possible to customize the JIRA dashboard manually or use ready-made templates
6. It is possible to extend JIRA capabilities and fine-tune it for the team’s usage via external plugins and JIRA add-ons. It is also possible to customize the JIRA dashboard manually or use ready-made templates.

2.6 Xray by Xblend

Xray is JIRA-based software for test management. Xray integrates JIRA functionality in addition to its planning, design, execution, and reporting of manual and automated testing. Xray can be used for Agile development. Test specification steps in Xray are attachment of test steps, links to defects and requirements, precondition for test cases, reusing the already existing preconditions for testing. Execution of tests in Xray includes creating, editing, scheduling test executions, test run and test step statuses customization. Additional automation of company specific processes is possible through Xray's REST API.

Progress during and post testing through a tracking the progress bar and built-in advanced reporting. Efficient Xray workflow [21] is attached in Appendix B and C. The Kanban board in Xray gives immediate indication of coverage status of the user stories, progress of Test Plans, progress of Test Executions, this can be seen in Figure 8. Tests can be filtered by their type or a JIRA specific JQL can be used to filter for keywords.


Sprint 1

QUICK FILTERS: Only My Issues Recently Updated

To Do

CALC-838
As a user, I can calculate the sum of 2 numbers
v3.0 - NOTRUN

CALC-848
Sub Test Execution for CALC-838

CALC-954
Sub Test Execution for CALC-838

In Progress

CALC-970
plan for v3.0, sprint 1

CALC-962
As a user, I can calculate the multiplication of 2 numbers
v3.0 - OK

CALC-966
Sub Test Execution for CALC-962

CALC-967
As a user, I can subtract 2 numbers
v3.0 - NOK

CALC-969
Sub Test Execution for CALC-967

Done

~~CALC-961~~
As a user, I can calculate the division of 2 numbers
v3.0 - OK

~~CALC-964~~
Sub Test Execution for CALC-961

Figure 8. Xray Kanban board example [22]

2.7 Microsoft Azure DevOps

Software by Microsoft with ability to track full testing life cycle. Tester must be individually assigned to one of the three available levels of tests:

- Test case level
- Planning testing/Preparing the test suites
- Test execution

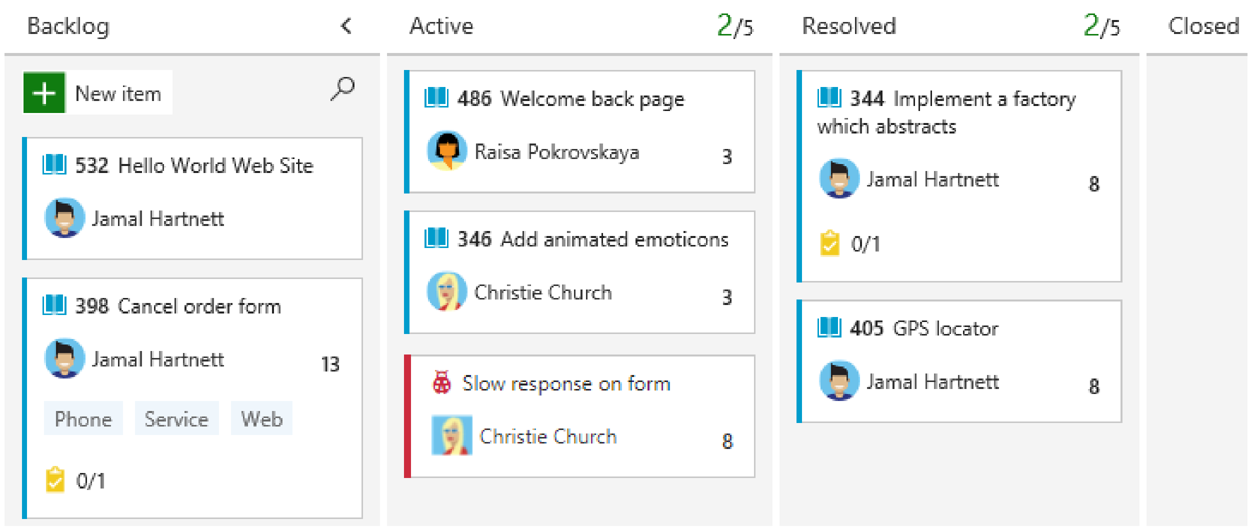
Table below explains the keywords of work with Microsoft Azure DevOps.

Table 2. Microsoft Azure DevOps keyword description [23]

Azure Boards hub	**Functions**
Work items	Access lists of work items based on specific criteria, such as those assigned to you, ones you follow, and work items you viewed or updated.
Boards	View work items as cards and update their status through drag-and-drop, similar to physical sticky notes on a whiteboard. Use this feature to implement Kanban practices and visualize work flow for a team.
Backlogs	view, plan, order, and organize work items, including using a product backlog to represent your project plan and a portfolio backlog to group work under features and epics.
Sprints	access your team's filtered view of work items based on a specific sprint or iteration path. Assign work to a sprint using drag-and-drop from the backlog. Interact with a backlog list or card-based taskboard to implement Scrum practices.
Queries	Generate custom work item lists and perform various tasks, such as triage work, make bulk updates, and view relationships between work items. Queries also allow for creating status and trend charts that can be added to dashboards.
Delivery Plans	Management teams can view deliverables and track dependencies across multiple teams in a calendar view. Delivery plans support tasks such as viewing up to 15 team backlogs, custom portfolio backlogs and epics, and work that spans several iterations. Users can add backlog items to a plan, view rollup progress of features and epics, and dependencies between work items.

Test case modification history is available, to have the ability to go back to previous test case versions and trace the changes. According to Microsoft Azure DevOps documentation [24] it has a REST API, which provides endpoints for HTTP methods to automate processes or extract information for mass data wrangling on the test reports.

Azure Boards is a service that is provided within Azure DevOps. It is a customizable tool, used for visualization and workflow optimization. Azure Boards have all the components needed for agile development such as dashboard, backlog, user stories, reports, tasks, notifications. Example of an Azure board is attached below.



Backlog	Active 2/5	Resolved 2/5	Closed
New item			
532 Hello World Web Site; Jamal Hartnett	486 Welcome back page; Raisa Pokrovskaya; 3	344 Implement a factory which abstracts; Jamal Hartnett; 8; 0/1	
398 Cancel order form; Jamal Hartnett; 13; Phone Service Web; 0/1	346 Add animated emoticons; Christie Church; 3	405 GPS locator; Jamal Hartnett; 8	
	Slow response on form; Christie Church; 8		

Figure 9. Microsoft Azure Boards Kanban example [25]

It is possible to forecast the number of sprints it would take to finish work on a product, based on the velocity, which is the pace that the team finishes a sprint.

2.8 TEST-GUIDE by TraceTronic

While the TEST-GUIDE tool doesn't have an implemented Kanban board, it has a dashboard with current run tests and the number of errors/successes and the execution rate of the test.

TEST-GUIDE capabilities:

1. Planning test coverage based on release
2. Requirements for testing are setup so that the conditions and test configuration is visible to the user.
3. Supports traceability

4. Tracks test case coverage (useful with the small changes and differences that are introduced in new releases)
5. Makes continuous integration possible by implementing a pipeline to automate the testing process

TEST-GUIDE workflow is attached below on Figure 10.

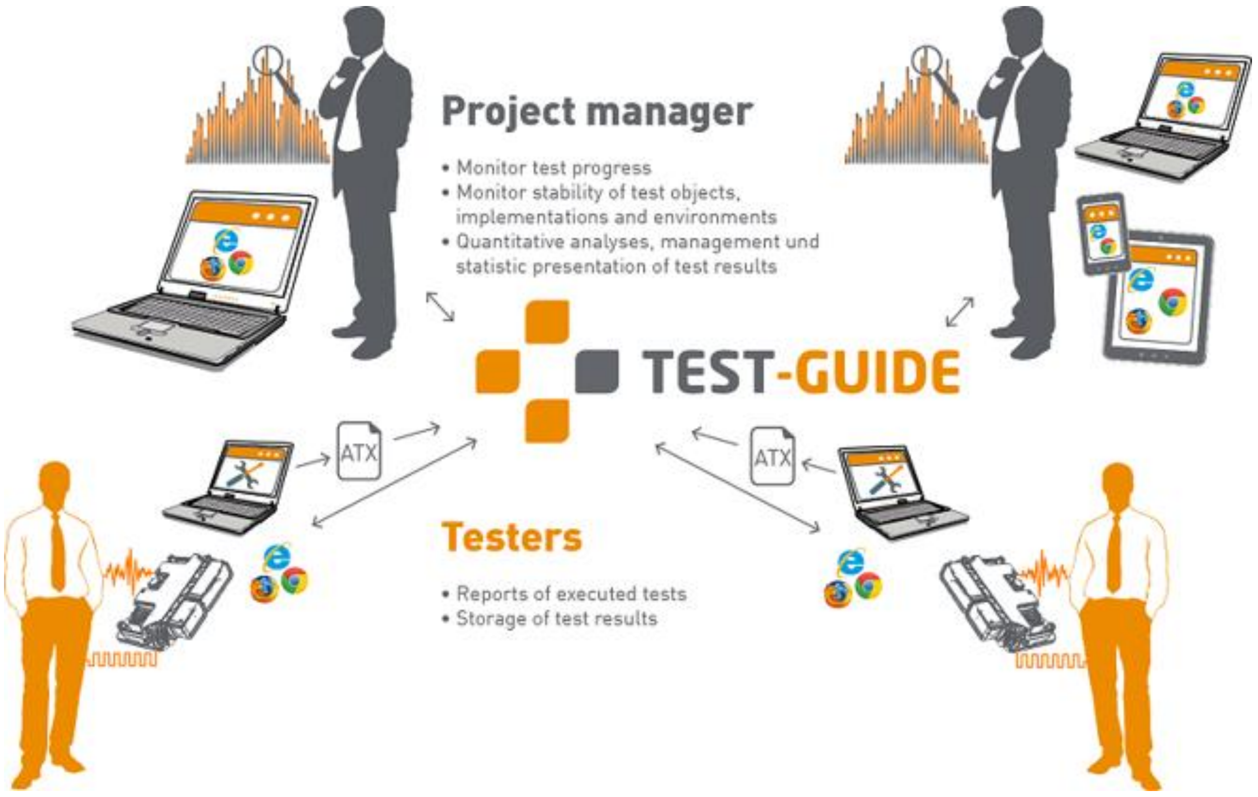

Project manager
• Monitor test progress
• Monitor stability of test objects, implementations and environments
• Quantitative analyses, management und statistic presentation of test results
TEST-GUIDE
ATX
ATX
Testers
• Reports of executed tests
• Storage of test results

Figure 10. TEST-GUIDE workflow [26]

Key difference between the previous software types and TEST-GUIDE lies in the playbooks. Playbook is like a test suite, that contains the configuration (requirements on the software and HiL side, test steps) that the test cases inside of it need to run. Playbooks are immutable, which means that once they are created, they cannot be changed, but newer versions of a playbook can be associated with the initial version. This can help track the history of changes between releases.

There is a test bench (HiL) activity distribution tab that distributes the testing to use up time more efficiently and reduce the time that HiL machines are idle without any test running.

Chapter 3. Test Management Tool Development

3.1 Baseline requirements for the tool

Based on an interview with a team leader Miguel Santiago Lasso Alvear, some of the test tool management prototype platforms are JIRA, TestGuide, X-Ray..

The interview to determine the baseline requirements was conducted with team leaders BSc. Daniel Tome Soares and Ing. Emil Minar. During the discussion basic terms and stages of the HiL testing were introduced and the requirements of the department were set. The following graph represents the requirements.

Description of the Kanban columns based on the Figure 10:

1. “Will come in the future”

Is a column containing the tests that are run on regular bases or that project leader expects to arrive. At this stage the test is visible to every team member and additional responsibilities like ordering software, cables, preparing the automation for the results and other necessities for test execution can be assigned. This column will have one of the following names in the end tool: “Pending Tests”, “Upcoming tests”.

2. “Request to test”

Is a column where the ticket is dragged once the data for the testing arrives. Will be renamed to “Requested”.

3. “Book slot for the playbook (if already exists)” and “Create playbook and book a slot (if the playbook doesn’t exist)”.

Booking the slot on one of the HiL machines in Porsche Engineering laboratory. If the playbook for the test already exists (on the tests run regularly) the configuration setup can be reused and only the data needs to be changed. While if the playbook doesn’t exist, configuration needs to be set up from scratch. This column might be merged and have background logic that checks if the playbook already exists and based on that either creates one or reuses the found one. These columns can also be kept as separate columns or have some marker for visibility.

4. “Check if running”

Ticket is moved to this column once the test starts running. There is a possibility to check the status of the test in a database. Current plan is to connect the database and track the status of the test, so that once the test starts it enters the “Check if running” column and leaves once the test execution on the HiL is finished.

5. “In review”

Once the testing ticket is done, it moves to this column and the ticket for a team member to review the result is assigned. Once the test results are reviewed, if the results match with the expectations, then the ticket is placed into “Delivered” column. If the test results are defective, the ticket status is adjusted to “Request the test”, which puts it to the corresponding column.

6. “Delivered”

Once the results are delivered to the customer and there is feedback on whether the test results are valid. If the results are valid then the ticket is closed. If not, the ticket status is adjusted to “Request the test” and the corresponding column.

Draft of the baseline Kanban board and description of functionality

Upcoming Tests	Requested Tests	Book the slot	Running Tests	In Review	Delivered	Closed

Figure 11. Draft of the Kanban Board

Once a new test appears, it is in the first column, to be prepared. Afterward, the test can be either dragged by the user or be automated, whilst being connected to the database and automatically move to a different column. Figure 12 describes the logic of the tool.

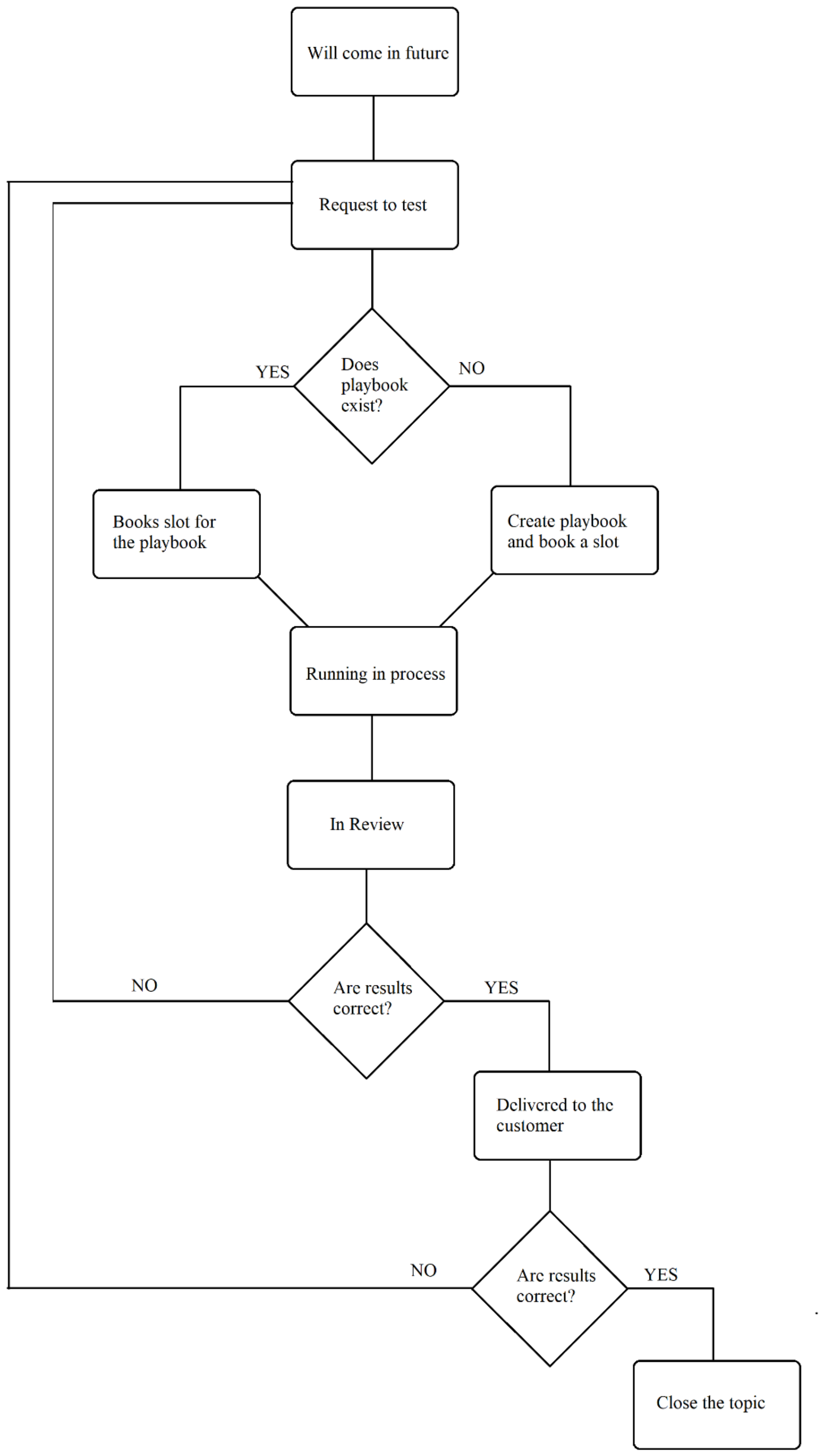


Figure 12. Outline of the tool functionality

3.2 Development platform choice

Initially, the tool was planned to be placed inside the Test Guide by Trace Tronic software either in the project-specific dashboards tab or as its own tab while letting the user pick the project filtering. Test Guide is commercially available license-based software without open-source code, which means that on-platform custom plugin development is not possible.

After a discussion with the team leader BSc. Daniel Tome Soares and the Test Guide technical support it was decided that the environment for the tool should be external and connected to the Test Guide's database. Trace Tronic technical support suggested using the webhooks or the REST API to retrieve or modify data stored inside the Test Guide software.

The development then was moved to a company-based Tool Integration System (TIS). TIS is a Django framework-based website.

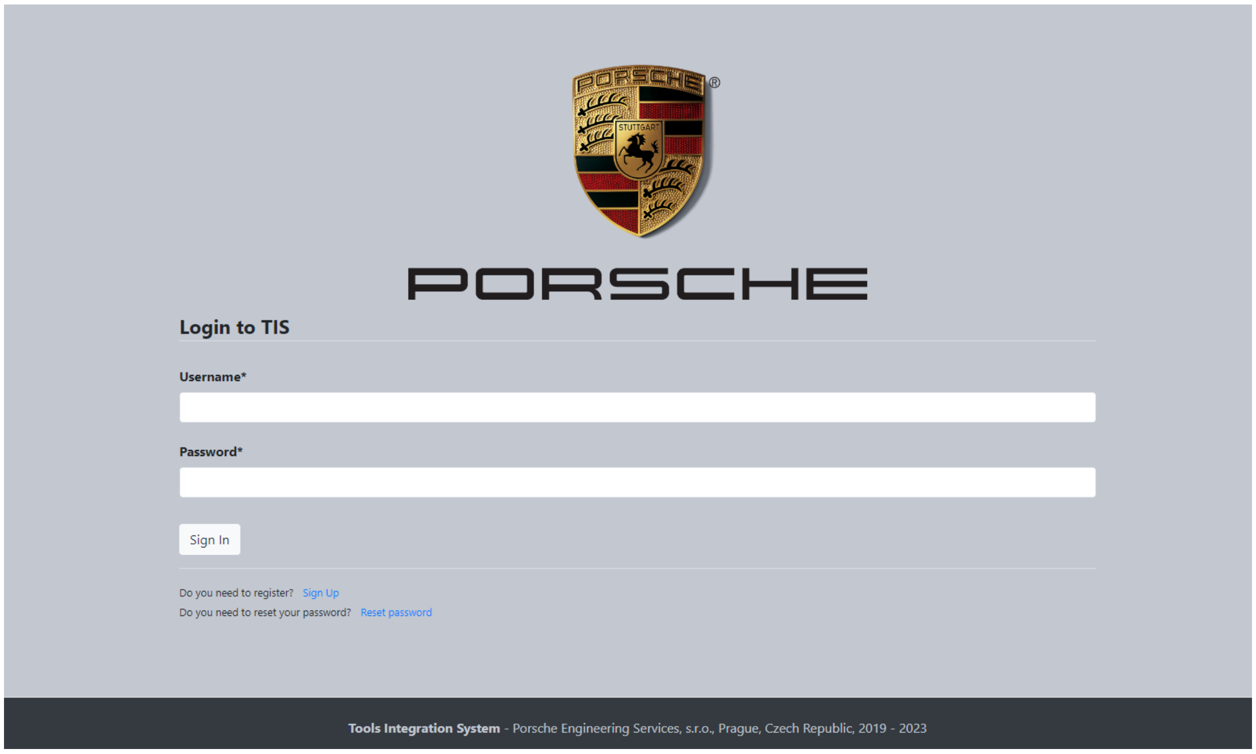

PORSCHE
Login to TIS
Username*
Password*
Sign In
Do you need to register? Sign Up
Do you need to reset your password? Reset password
Tools Integration System - Porsche Engineering Services, s.r.o., Prague, Czech Republic, 2019 - 2023

Figure 13. TIS landing page

TIS is fully customizable and has the means for integration of JavaScript framework called vue.js as well as usage of typescript and python web framework Django, which allows dynamic backend to frontend communication, storage of data in the SQLite based database, setting up custom CRUD APIs.

3.3 Test Guide database connection possibilities

APIs allow the communication between two applications and acts as a postman between them, retrieving, writing, or modifying the data sent from one app to another. This can be used for remote automation or processes and in our case, it will be used to load the test releases and test tasks into the TestGuide software from the TIS web site. It will also be used to track the status of the test running, create playbooks, schedule tests. So, for the tool we need an API that can have multiple HTTP request types and be versatile.

The two options of database connection provided by Trace Tronic are REST API and a webhook. REST API is standardized API that must be setup manually, while webhook is already setup, but only to send data to the user the moment it is modified, so it has only part of the functionality of an API, so usually webhooks can be referred to as a subset of APIs since they are more limited in their functionality.

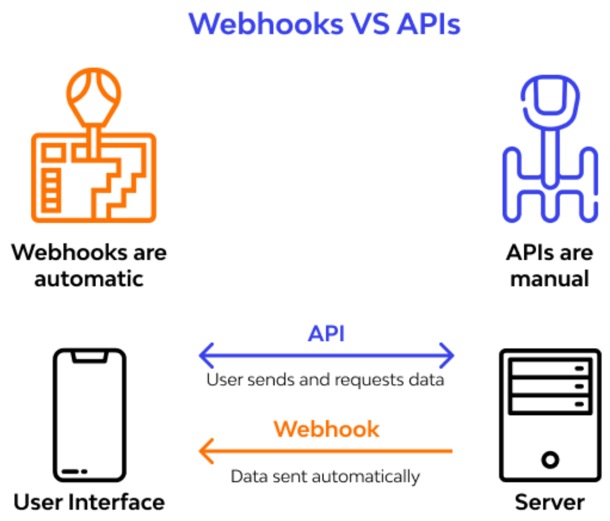


Figure 14. Difference in operation between webhooks and APIs [27]

Therefore, since there are several usages of different API requests in the tool, REST API is a better suited option.

Test Guide API is based on request links. The types of the sub-APIs are shown on the following Figure 15.

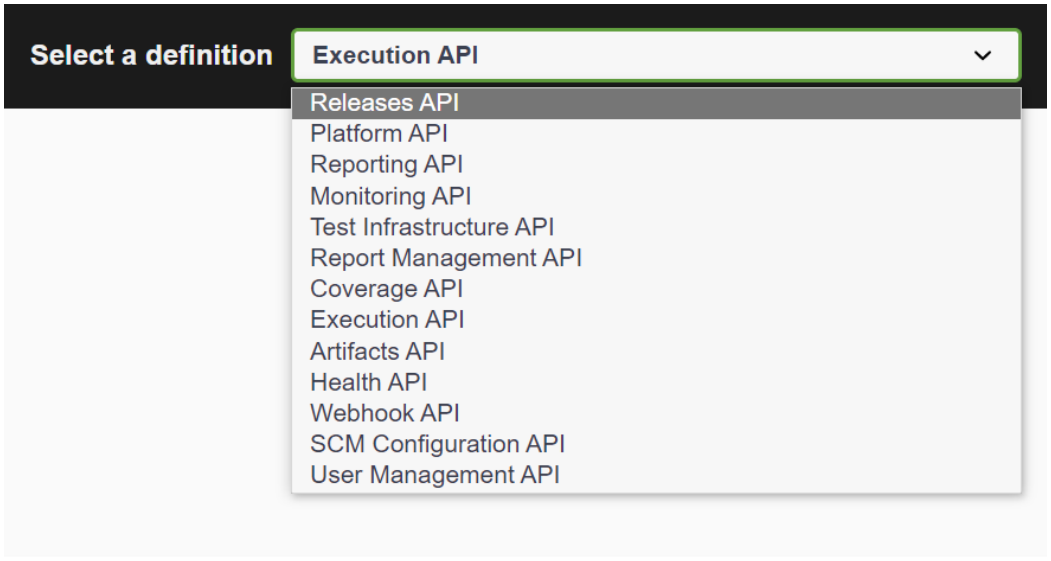


Figure 15. TEST-GUIDE REST API libraries

For the prototype following APIs are used:

- Releases API – for the test release creation from the task database.
- Execution API – for creating playbooks and executing them based on a request of running a playbook manually or once the execution time in the playbook is reached.
- Artifacts API – used to manage the status of the playbook, when it was last updated and modified.

3.4 TIS app setup for the prototype

TIS is built on python web-framework Django. In Django the app container needs to be setup first.

`content.html`

Front end of the tool prototype is in `content.html`, it contains the page name and connection hook to vue.js file that has main logic of the front end.

Front end of the TIS website is contained in the `templates.html` file. To see the app on the website, the `content.html` file is then connected to `templates.html` as an extension using the following command.

```
{% extends "template.html" %}
```

apps.py

The setup for the app container is connected to an already running website, the app configuration is specified in apps.py script. Parameters set at this step are the name of the tool, URL prefix and apps contribution to the side menu.

```python
from django.apps import AppConfig
from tis.navigation import side_menu

class TestOverviewAppConfigClass(AppConfig):
    """ Class sets the name, url, and toolbar specifications
      (color, icon, positioning)  """
    name = 'modules.test_management_overview'

    url_prefix = 'test_overview_tool'

    side_menu.contribute(
        {
            'item_type': 'nav-link',
            'name': 'Test Overview',
            'link': 'kanban-overview',
            'icon': 'fas fa-th',
            'parent': 'Common Tools',
            'position': 11,
        },
    )
```

views.py

forms.py

In `forms.py` the data from front end is submitted to backened and is handled in `views.py.` In views.py the backend logic of the script is setup and other python scripts for automation can be connected at this point. However, the main logic of the test management tool will be contained in the vue.js script, so the `views.py` and `forms.py` script was setup to render the default state of the page without any other functionality.

3.5 TIS CRUD API

models.py

A model in Django is an element in an SQLite based database. Each model can be assigned fields which are columns in a table. Each model instantiated in Django is saved as a row in the table.

Task database is for task cards that are shown in the Kanban board.

Table 3.Contents of the Task database

TASK DATABASE
project_id = models.IntegerField()
project_name = models.CharField(max_length=1000)
playbook_id = models.IntegerField()
release_name = models.CharField(max_length=1000)
vehicle = models.CharField(max_length=1000)
hil = models.CharField(max_length=1000)
dspace_model = models.CharField(max_length=1000)
architecture = models.CharField(max_length=1000)
software = models.CharField(max_length=1000)
specifications = models.CharField(max_length=1000)
time = models.CharField(max_length=1000)
state = models.CharField(max_length=100)
data = models.FileField()

Playbook database is used to create/load playbooks from the Test Guide database to automate the test execution process.

Table 4. Contents of playbook database

PLAYBOOK DATABASE
project_name = models.CharField(max_length=1000)
playbook_id = models.IntegerField()
project_id = models.IntegerField()
name = models.CharField(max_length=1000)
type = models.CharField(max_length=1000)
url = models.CharField(max_length=1000)
commit = models.CharField(max_length=1000)
relative_path = models.CharField(max_length=1000)
label = models.CharField(max_length=1000)
setup_steps = models.CharField(max_length=100000)
testcases = models.CharField(max_length=100000)
teardown_steps = models.CharField(max_length=100000)
xil_config_reqs = models.CharField(max_length=100000)

serializers.py

Is used to serialize the data from front end to backend and vice-versa to pass it as a JSON type of response.

urls.py (Django project, not the app)

Is setup to transmit the GET response of the API to get data about the tasks and put it into the Kanban board.

In the Django framework, there are various options available for setting up APIs. One common approach is to use Django REST framework, which provides a standardized and robust API solution. With Django REST framework, we can define API endpoints, serializers to convert data models into JSON or other formats, and views to handle incoming API requests. This manual setup ensures flexibility and customization options tailored to our specific needs.

admins.py

Allows to use the Django admin interface to monitor the API state. This page was setup for both databases.

views.py (TIS CRUD API request setup)

```
from modules.test_management_overview.tis_api.serializers
import TasksSerializer
from modules.test_management_overview.models import Tasks
from rest_framework.parsers import JSONParser
from rest_framework.response import Response
from rest_framework.views import APIView

class TasksAPI(APIView):
    def get(self, request, format=None):
        cards = Tasks.objects.all()
        serializer = TasksSerializer(cards, many=True)
        return Response(serializer.data)

    def post(self, request, format=None):
        card_data = JSONParser().parse(request)
        serializer = TasksSerializer(card_data, many=True)
        if serializer.is_valid():
            serializer.save()
        return Response(serializer.data)
```

GET – retrieves the data on the backend database, so it picks up the information about the task card.

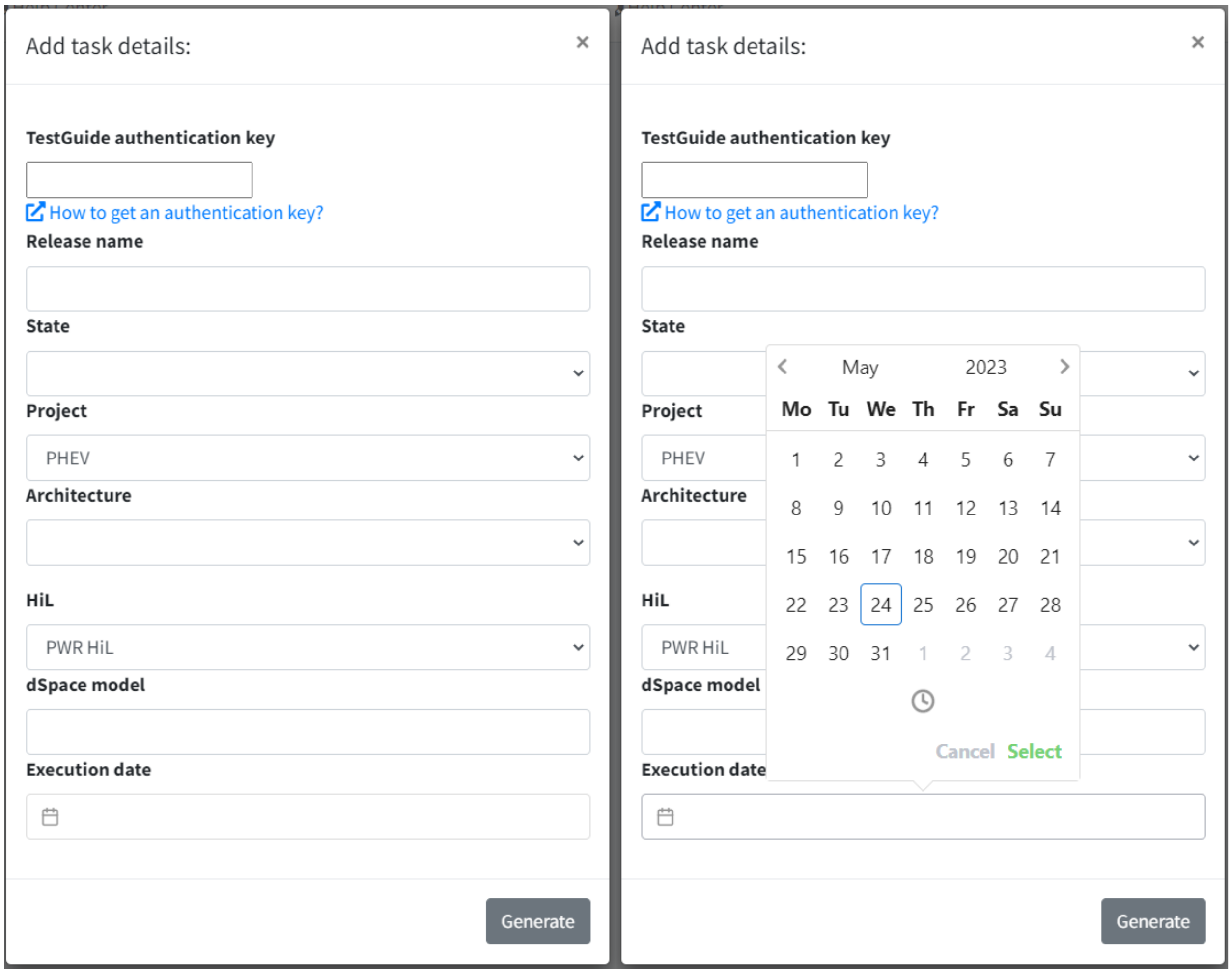
POST – puts data inside the database when the user creates a new task.

3.6 Front end implementation and tool logic

“Add new task” button is used to populate the board with tasks.

TEST-GUIDE authentication key is required by the TEST-GUIDE API as a safety measure. This key is generated inside the user cabinet.

Other fields are necessary to fill out to populate the data inside a task card model and then the playbook for test execution.



Add task details:

TestGuide authentication key

How to get an authentication key?

Release name

State

Project: PHEV

Architecture

HiL: PWR HiL

dSpace model

Execution date

Generate

Add task details:

TestGuide authentication key

How to get an authentication key?

Release name

State

Project: PHEV

Architecture

HiL: PWR HiL

dSpace model

Execution date

Generate

May 2023

Mo	Tu	We	Th	Fr	Sa	Su
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	1	2	3	4

Cancel Select

Figure 16. On the left is the default state of the "Add task" form. On the right a 'vue-datepicker' library usage example for choosing the time of test execution.

Once a task card is retrieved with the asynchronous fetch function in vue.js file, the tasks are shown on the board.

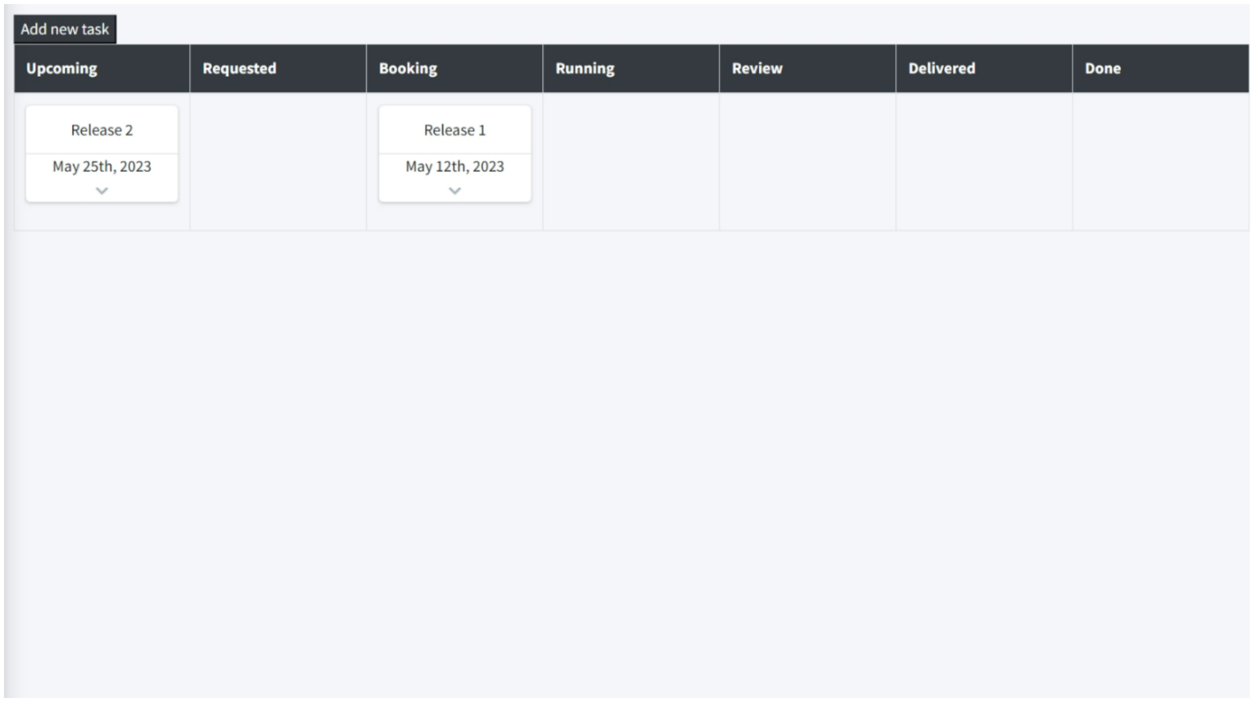


Figure 17. Initial state of the board

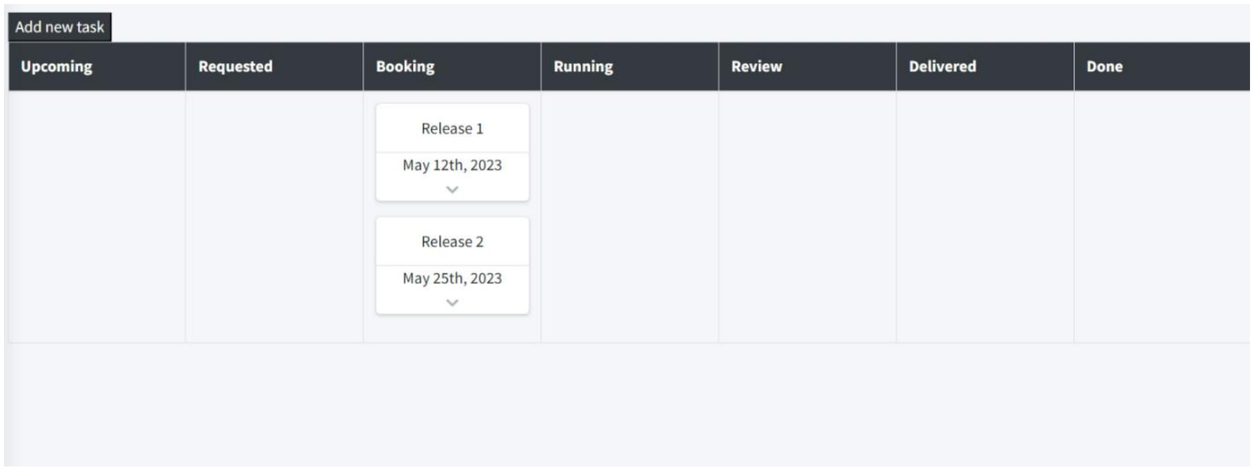


Figure 18. Automatic adjusting of the board borders depending on how many cards are stacked on top of each other

The images of the tool page are also in the Appendix for purposes of readability.

Conclusion

Theoretical study focused on the application of HiL (Hardware-in-the-Loop) testing, test management, and agile methodologies such as lean, Scrum, and Kanban. The adoption of lean principles, Scrum, and Kanban methodologies demonstrated their potential to streamline testing activities, enhance transparency, improve flexibility, productivity, collaborations, and promote continuous improvement.

Analysis of current practices at Porsche Engineering with team leaders of Chassis and Powertrain teams, insights of challenges of HiL test management were gained, emphasizing the need for development of a custom test management tool.

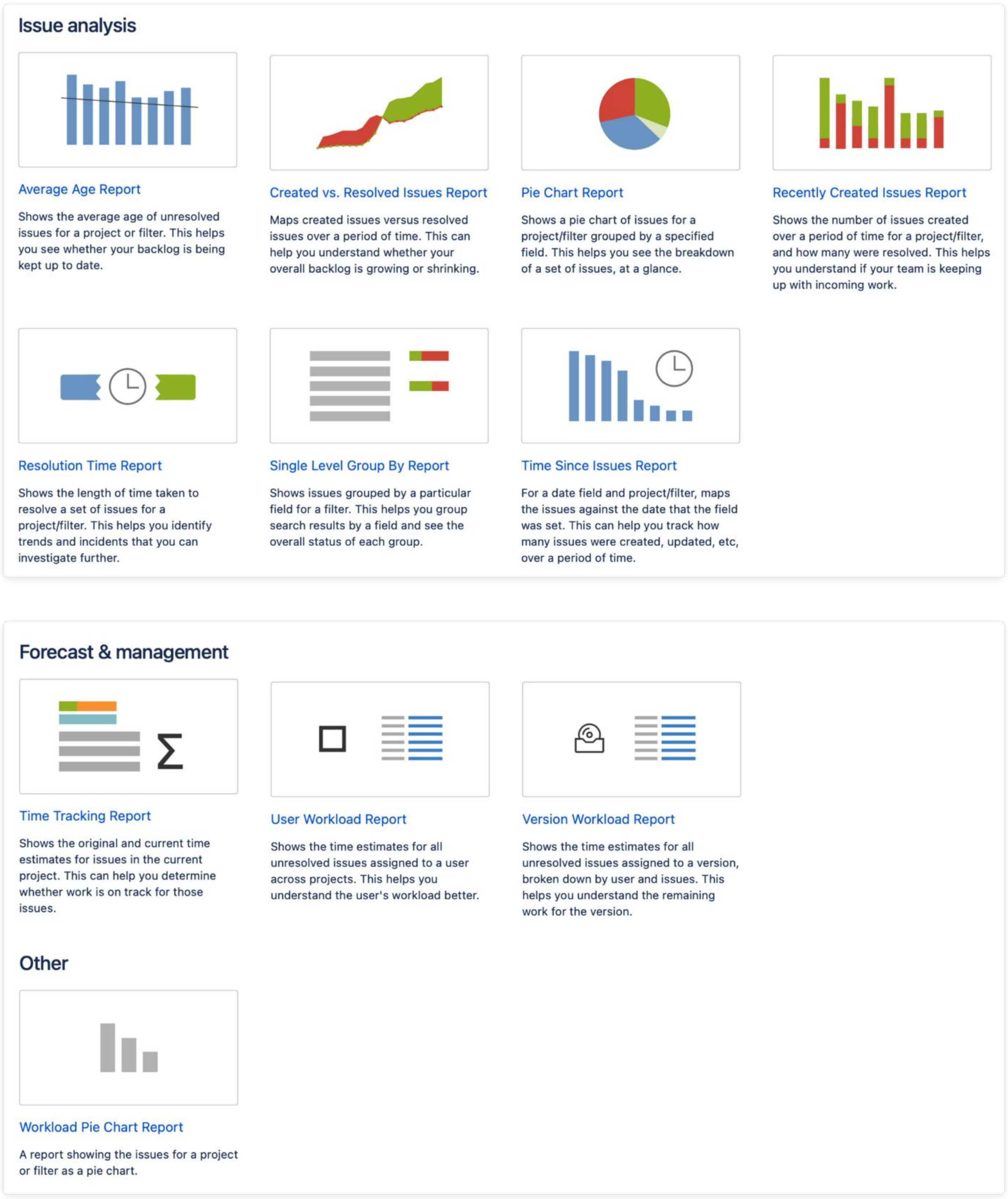
This thesis project developed a tool prototype on the Django-based company website, that effectively integrates with an external Test Guide API, providing a user-friendly web page featuring an interactive Kanban board with task cards. The implementation of this tool demonstrated the practical application of web technologies and API connectivity in facilitating task management and organization.

The Kanban board's intuitive interface allows users to visually track and manage their tasks, fostering collaboration and improving workflow efficiency. "Add new task" button enhances the tool's functionality by enabling users to populate the board with tasks.

In future, the tool's features in terms of automation will be enhanced. This work will be continued by another student employee of Porsche Engineering as a part of their master thesis.

List of Appendices

Appendix A. JIRA reporting options.



Issue analysis

Average Age Report

Shows the average age of unresolved issues for a project or filter. This helps you see whether your backlog is being kept up to date.

Created vs. Resolved Issues Report

Maps created issues versus resolved issues over a period of time. This can help you understand whether your overall backlog is growing or shrinking.

Pie Chart Report

Shows a pie chart of issues for a project/filter grouped by a specified field. This helps you see the breakdown of a set of issues, at a glance.

Recently Created Issues Report

Shows the number of issues created over a period of time for a project/filter, and how many were resolved. This helps you understand if your team is keeping up with incoming work.

Resolution Time Report

Shows the length of time taken to resolve a set of issues for a project/filter. This helps you identify trends and incidents that you can investigate further.

Single Level Group By Report

Shows issues grouped by a particular field for a filter. This helps you group search results by a field and see the overall status of each group.

Time Since Issues Report

For a date field and project/filter, maps the issues against the date that the field was set. This can help you track how many issues were created, updated, etc, over a period of time.

Forecast & management

Time Tracking Report

Shows the original and current time estimates for issues in the current project. This can help you determine whether work is on track for those issues.

User Workload Report

Shows the time estimates for all unresolved issues assigned to a user across projects. This helps you understand the user's workload better.

Version Workload Report

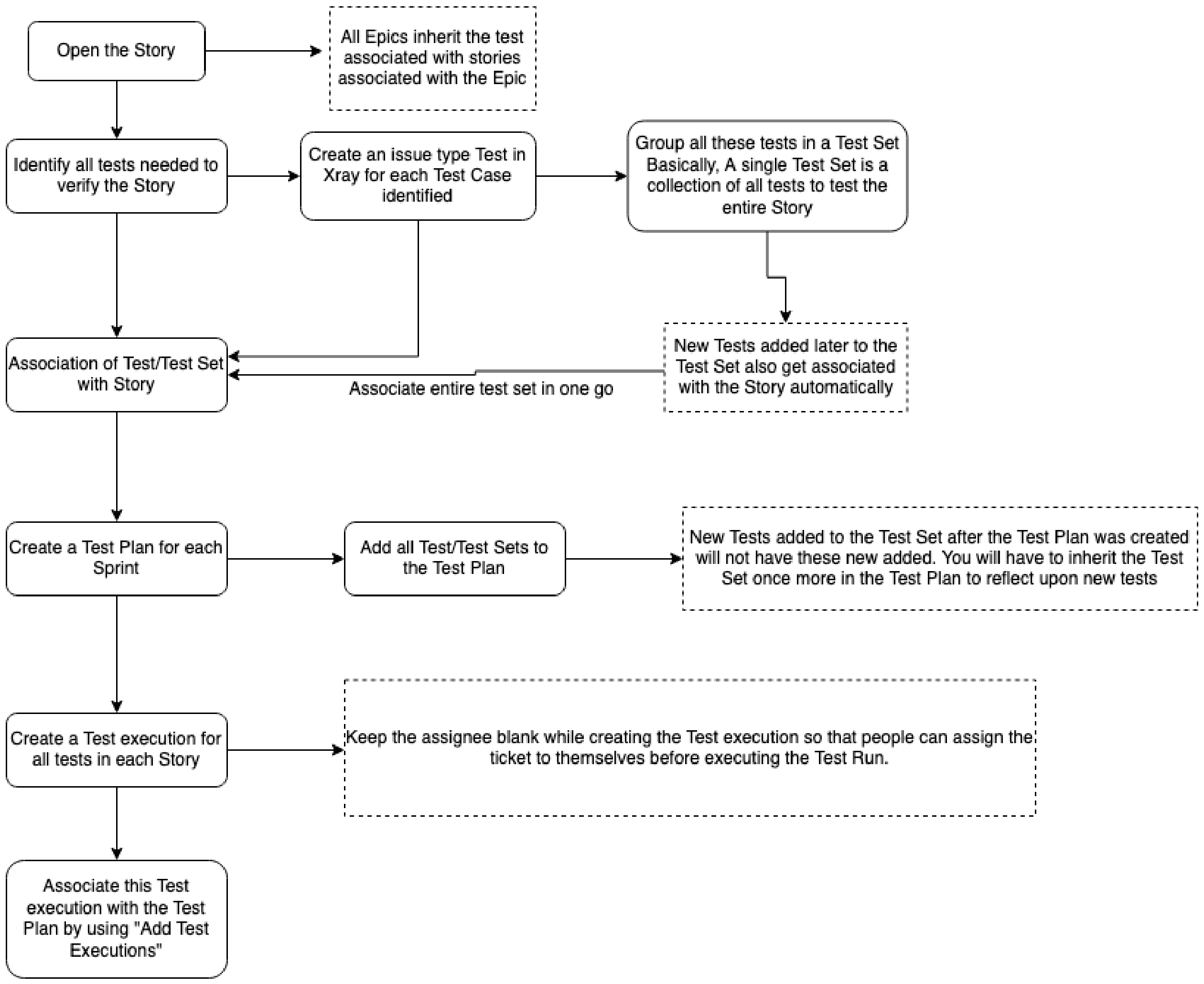
Shows the time estimates for all unresolved issues assigned to a version, broken down by user and issues. This helps you understand the remaining work for the version.

Other

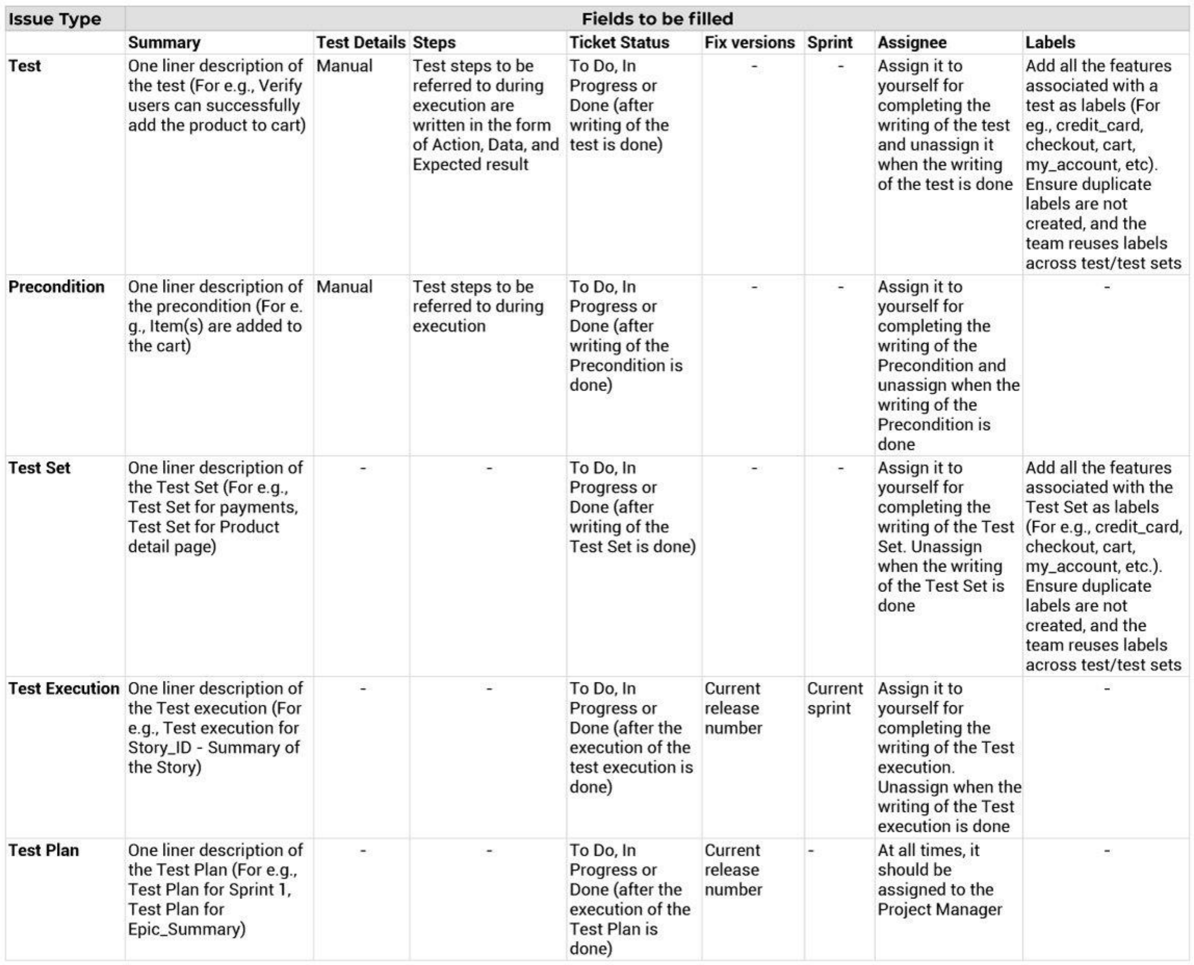
Workload Pie Chart Report

A report showing the issues for a project or filter as a pie chart.

Appendix B. Xray workflow example

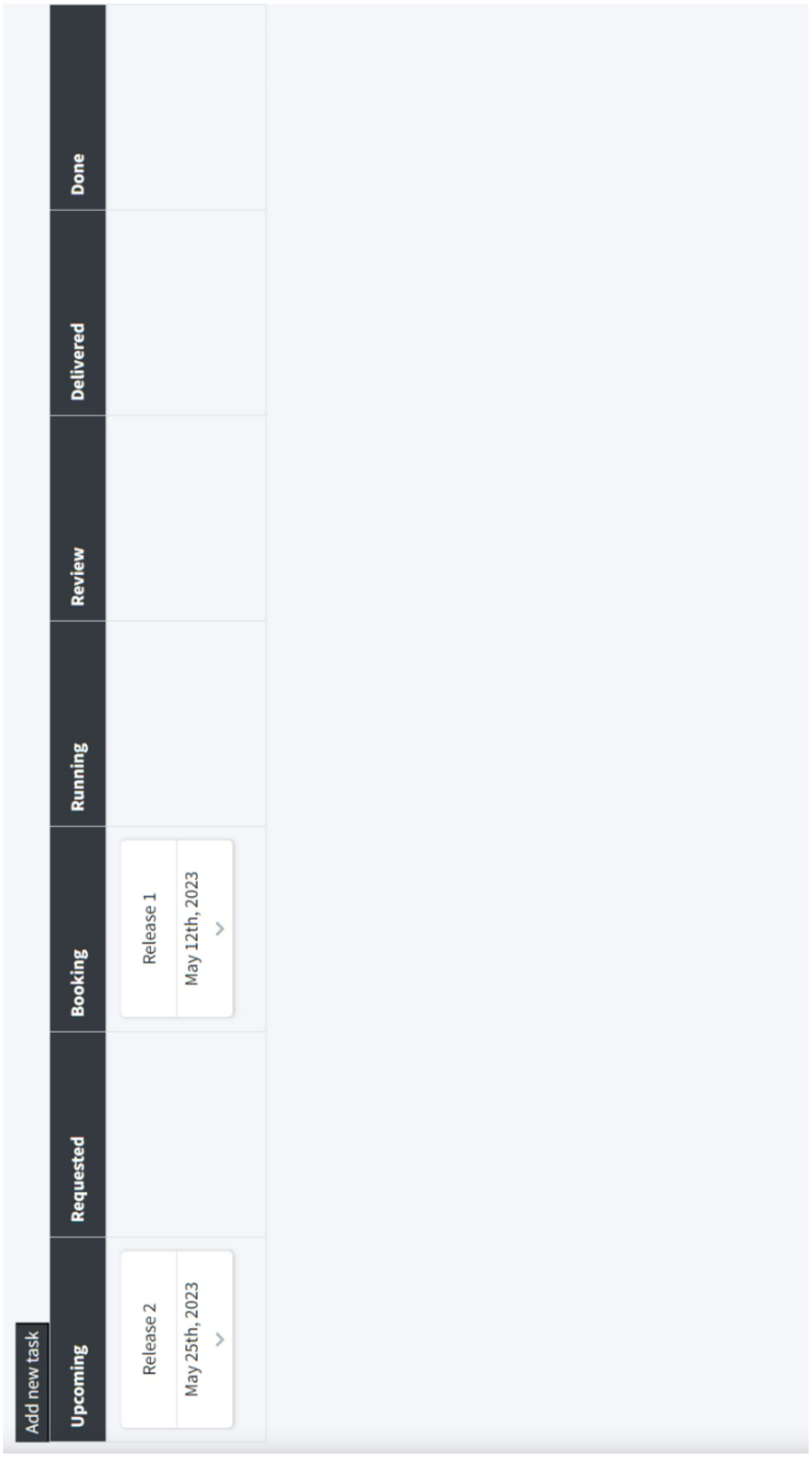

Open the Story
All Epics inherit the test associated with stories associated with the Epic
Identify all tests needed to verify the Story
Create an issue type Test in Xray for each Test Case identified
Group all these tests in a Test Set Basically, A single Test Set is a collection of all tests to test the entire Story
Association of Test/Test Set with Story
Associate entire test set in one go
New Tests added later to the Test Set also get associated with the Story automatically
Create a Test Plan for each Sprint
Add all Test/Test Sets to the Test Plan
New Tests added to the Test Set after the Test Plan was created will not have these new added. You will have to inherit the Test Set once more in the Test Plan to reflect upon new tests
Create a Test execution for all tests in each Story
Keep the assignee blank while creating the Test execution so that people can assign the ticket to themselves before executing the Test Run.
Associate this Test execution with the Test Plan by using "Add Test Executions"

Appendix C. Xray Issue Type for Appendix B workflow.



Issue Type	Fields to be filled							
	Summary	Test Details	Steps	Ticket Status	Fix versions	Sprint	Assignee	Labels
Test	One liner description of the test (For e.g., Verify users can successfully add the product to cart)	Manual	Test steps to be referred to during execution are written in the form of Action, Data, and Expected result	To Do, In Progress or Done (after writing of the test is done)	-	-	Assign it to yourself for completing the writing of the test and unassign it when the writing of the test is done	Add all the features associated with a test as labels (For eg., credit_card, checkout, cart, my_account, etc). Ensure duplicate labels are not created, and the team reuses labels across test/test sets
Precondition	One liner description of the precondition (For e. g., Item(s) are added to the cart)	Manual	Test steps to be referred to during execution	To Do, In Progress or Done (after writing of the Precondition is done)	-	-	Assign it to yourself for completing the writing of the Precondition and unassign when the writing of the Precondition is done	-
Test Set	One liner description of the Test Set (For e.g., Test Set for payments, Test Set for Product detail page)	-	-	To Do, In Progress or Done (after writing of the Test Set is done)	-	-	Assign it to yourself for completing the writing of the Test Set. Unassign when the writing of the Test Set is done	Add all the features associated with the Test Set as labels (For e.g., credit_card, checkout, cart, my_account, etc.). Ensure duplicate labels are not created, and the team reuses labels across test/test sets
Test Execution	One liner description of the Test execution (For e.g., Test execution for Story_ID - Summary of the Story)	-	-	To Do, In Progress or Done (after the execution of the test execution is done)	Current release number	Current sprint	Assign it to yourself for completing the writing of the Test execution. Unassign when the writing of the Test execution is done	-
Test Plan	One liner description of the Test Plan (For e.g., Test Plan for Sprint 1, Test Plan for Epic_Summary)	-	-	To Do, In Progress or Done (after the execution of the Test Plan is done)	Current release number	-	At all times, it should be assigned to the Project Manager	-

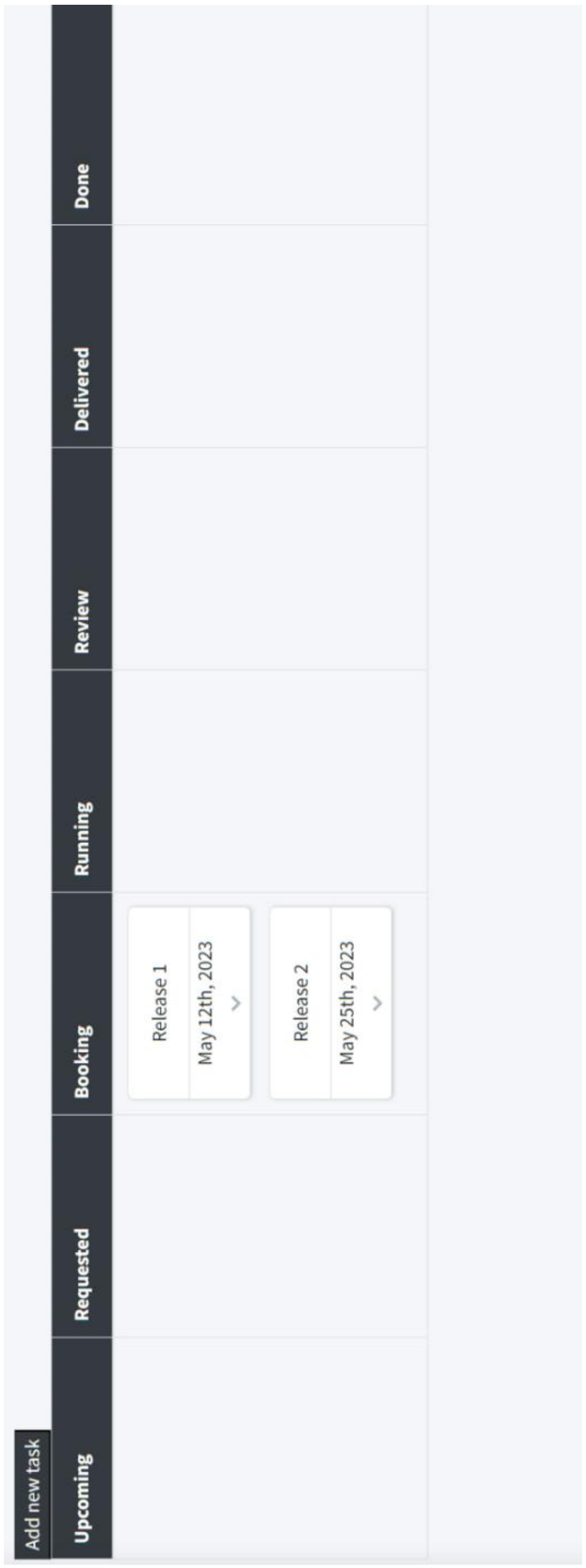
Appendix D. Initial State of the Board



Add new task

Upcoming	Requested	Booking	Running	Review	Delivered	Done
Release 2 May 25th, 2023		Release 1 May 12th, 2023				

Appendix E. Board After Shifting Release 2 to the Booking Column



List of references

[1] H. Tummescheit and C. Haugstetter, "Caltech. Hardware-in-the-loop Simulation & Analysis," 2003. [Online]. Available: http://www.cds.caltech.edu/archive/help/uploads/wiki/files/12/HIL-DGC100.pdf. [Accessed 3rd March 2023].

[2] G. Yadav, Y. Liao and A. D. Burfield, "Hardware-in-the-Loop Testing for Protective Relays Using Real Time Digital Simulator (RTDS)," *Energies*, vol. 16, no. no. 3, p. 1039, 2023.

[3] G. Mike, "An Introduction to the Cost of Change and Technical Debt," 12 November 2015. [Online]. Available: https://www.projectmanagement.com/articles/308195/An-Introduction-to-the-Cost-of-Change-and-Technical-Debt. [Accessed 6 March 2023].

[4] L. v. Dijk, "Future Vehicle Networks and ECUs. Architecture and Technology considerations.," 2017. [Online]. Available: https://www.nxp.com/docs/en/white-paper/FVNECUA4WP.pdf. [Accessed 4 March 2023].

[5] ""What is Hardware-in-the-loop"," National Instruments, 6 March 2023. [Online]. Available: https://www.ni.com/en/solutions/transportation/hardware-in-the-loop/what-is-hardware-in-the-loop-.html.

[6] A. Atar, Hands-on test management with Jira : end-to-end test management with Zephyr, synapseRT, and Jenkins in Jira, Birmingham : Packt Publishing, 2019.

[7] L. Williams, "Agile Software Development Methodologies and Practices," in *Advances in Computers*, vol. 80, M. V. Zelkowitz, Ed., Elsevier, 2010, pp. 1-44.

[8] S. S, "What Is Agile Project Management? Definition & Agile Methodologies.," project-management.info, [Online]. Available: https://project-management.info/what-is-agile-project-management-definition-agile-methodologies/.

[9] A. Cockburn, "Agile Software Development," in *The Agile Software Development Series*, vol. 177, Boston, Addison-Wesley Longman Publishing Co., Inc., 2002, pp. 14-200.

[10] M. Cohn, Agile Estimating and Planning, Upper Saddle River, NJ, USA: Prentice Hall, 2005.

[11] J. Sutherland and J. J. Sutherland, Scrum: The Art of Doing Twice the Work in Half the Time, Crown, 2014.

[12] S. M. A. Z. N. B. Boban Melović, "MATEC Web of Conferences 86, 05029," in *The role of the concept of LEAN management in modern business*, 2016.

[13] "Essence of Lean," LEANPRODUCTION, [Online]. Available: https://www.leanproduction.com/essence-of-lean/. [Accessed 1 March 2023].

[14] D. J. Lu and N. N. K. , Kanban just-in-time at Toyota : management begins at the workplace / edited by the Japan Management Association ; translated by David J. Lu ; foreword by Norman Bodek., Rev. ed. ed., Cambridge (Mass.) : Productivity Press, 1989., 1989, pp. 7-26.

[15] "Agile Board design – the theory and practice of swimlanes," 20 February 2017. [Online]. Available: https://agilefixer.com/2017/02/20/agile-board-design-the-theory-and-practice-of-swimlanes/. [Accessed 9 April 2023].

[16] D. J. Anderson, Kanban: Successful Evolutionary Change for Your Technology Business, Blue Hole Press, 2010, pp. 27-129.

[17] H. Kniberg, Lean from the Trenches: Managing Large-Scale Projects with Kanban, Pragmatic Bookshelf, 2011, pp. 19-62.

[18] S. Saleh, "jexo.io," Appfire, 24 January 2022. [Online]. Available: https://jexo.io/blog/beginners-guide-to-jira/. [Accessed 1 March 2023].

[19] G. Siddharth , "How to follow the Kanban process for software development using JIRA," 18 December 2017. [Online]. Available: https://www.srijan.net/resources/blog/setting-up-kanban-board-in-jira. [Accessed 7 April 2023].

[20] "Kanban," Atlassian, [Online]. Available: https://www.atlassian.com/software/jira/templates/kanban. [Accessed 1 March 2023].

[21] S. Sharma, "How To Use Xray Test Management Tool In Agile Environment," Axelerant, 29 May 2020. [Online]. Available: https://www.axelerant.com/blog/how-use-xray-test-management-tool-agile-environment. [Accessed 1 March 2023].

[22] Xblend, "Agile Board Enhancements, Xray documentation," [Online].

[23] "What is Azure Boards?," Microsoft, 22 March 2023. [Online]. Available: https://learn.microsoft.com/en-us/azure/devops/boards/get-started/what-is-azure-boards?view=azure-devops. [Accessed 7 April 2023].

[24] "Azure REST API reference," Microsoft, 4 April 2023. [Online]. Available: https://learn.microsoft.com/en-us/rest/api/azure/. [Accessed 7 April 2023].

[25] "Boards and Kanban documentation," Microsoft, 2 February 2023. [Online]. Available: https://learn.microsoft.com/en-us/azure/devops/boards/boards/kanban-quickstart?view=azure-devops. [Accessed 7 April 2023].

[26] TraceTronic, "Ecu Test Version Release History," [Online]. Available: https://www.tracetronic.com/products/ecu-test/older-versions/.

[27] "Webhooks and REST," [Online]. Available: https://www.wallarm.com/what/a-simple-explanation-of-what-a-webhook-is.