LAMK Lahden ammattikorkeakoulu
Lahti University of Applied Sciences

Improving Scrum with Test-Driven Development

Case study: EasyScrum and EasyChat application

LAHTI UNIVERSITY OF APPLIED SCIENCES
Degree programme in Business Information Technology
Bachelor’s Thesis
Spring 2018
Gia Khoa Bui

Lahti University of Applied Sciences
Degree Programme in Business Information Technology

BUI, GIA KHOA:

Title: Improving Scrum with Test-Driven Development
Case study: EasyScrum and EasyChat application

Bachelor's Thesis in Business Information Technology

53 pages, 2 pages of appendices

Spring 2018

ABSTRACT

Over the past years, agile development methods have become popular, and Scrum is the most popular of these. One of the most used techniques to ensure success in Scrum is test-driven development - a software development principle that helps to attain high-quality production code.

The thesis presents two case studies. EasyScrum is an application that only applies the Scrum management framework, whereas EasyChat is an application that applies the test-driven development and Scrum. EasyScrum is a Scrum toolkit that helps teams in their agile workflow, and during the development process, several limitations of Scrum may occur. Therefore, EasyChat, a real-time chat application, was developed with test-driven development as a support technique to Scrum.

The main objective of this thesis is to identify the benefits of implementing test-driven development with Scrum by comparing the development process of two applications. For that, this thesis adopts deductive research approach and use qualitative research methods and collect data by applying observation.

By applying test-driven development during the development process, the author was able to see improvements in time usage and quality. The study shows test-driven development helps to save time and improves the quality of created the application and the code is more maintainable and flexible.

Keywords: agile development methodologies, Scrum, test-driven development, testing, EasyScrum, EasyChat.

CONTENTS

LIST OF ABBREVIATIONS

API	Application Programming Interface
CVCS	Centralized Version Control System
DVCS	Distributed Version Control System
HTML	Hypertext Mark-up Language
HTTP	Hypertext Transfer Protocol
LVCS	Local Version Control System
TDD	Test-Driven Development
URL	Uniform Resource Locator
VCS	Version Control System

LIST OF FIGURES

1 INTRODUCTION

1.1 Research background

According to VersionOne, agile development is a general term for multiple software development methodologies based on iterative development. If compared to the Waterfall development model, all agile development methodologies are lightweight, and they all support and enhance project management, collaboration, continuous development and not only allow fast delivery but ensure the high quality of products. (VersionOne 2018, 2-10.)

VersionOne publishes an annual report that shows the state of agile and the use of agile methodologies in organizations and enterprises throughout the world. According to the summary report, agile development adoption is growing at an accelerated rate among organization in every industry, both respondents from software companies and non-software companies increased significantly. Despite the increase, the survey also states that there is still room for growth. More and more organizations adopt agile development methodologies in their development process, and 94% of the respondents indicated practicing Agile. The report also highlights that Extreme Programming (XP), Scrum, Crystal, Dynamic Systems Development Method (DSDM), Lean Development, and Feature-Driven Development (FDD) are the most popular agile methodologies. Among these practices, Scrum is the most common agile method in the respondents’ organizations. (VersionOne 2018, 2-10.)

According to the latest annual State of the Agile™ survey, 98% percent of the respondents have benefited from agile methodologies and Scrum in particular (VersionOne 2018, 2). However, Scrum, the most widely used software management process, has several drawbacks. For example, since every sprint in a Scrum-based development process is only 2 to 4 weeks, product quality management can be a challenge. This characteristic of Scrum requires the development team to have an efficient development, communication and testing plan for the created software.

Due to such short sprints, developing features may drop late in the sprint cycle and make the developers lack time to do testing. Also, developers often end up spending more time on testing, debugging and retesting and lack time to review the features delivered to the client. (Sherman 2017.)

In software development, test plans play a significant role. Forming a test plan depends on several factors, for instance, the team's structure and maturity. Different organizations and teams will have different choices of testing methods (Mehra 2016). One of the most popular techniques that support testing is test-driven development (TDD).

Test-driven development, a software development technique, is very often associated with agile development methods. Several teams and organizations have implemented test-driven development technique to attain a high quality of the design and code. (Agarwal 2018.) Test-driven development will be explained in more detail in chapter 3. This thesis project was set to develop a chat application by adopting the Scrum framework and the test-driven development technique. The aim was to find out the benefits of applying TDD in the development process in comparison to only applying Scrum.

1.2 Thesis objectives and research question

This thesis aims to identify the benefits of implementing test-driven development with Scrum by comparing the development process of the EasyScrum application and EasyChat application. During the development process of the EasyScrum application, the author had encountered multiple problems and realized the Scrum framework has several limitations. The author focused on looking for a strategy or technique that can improve Scrum and discovered the test-driven development technique. Based on the finding, the EasyChat application was developed applied Scrum and test-driven development together to identify the benefits of TDD.

Consequently, the thesis aims to answer the following research question:

What are the benefits of the test-driven development and how could it be applied to improve Scrum?

1.3 Thesis structure

This thesis contains seven chapters. Figure 1 gives an overview of the main points of this thesis.

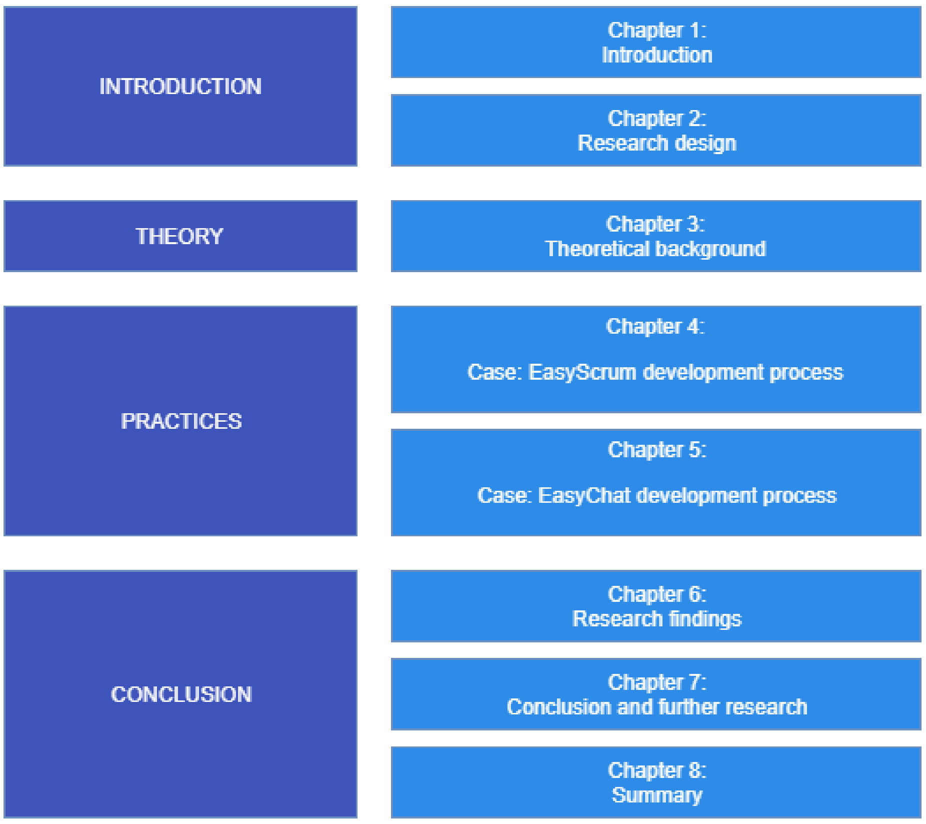

INTRODUCTION
Chapter 1: Introduction
Chapter 2: Research design
THEORY
Chapter 3: Theoretical background
PRACTICES
Chapter 4: Case: EasyScrum development process
Chapter 5: Case: EasyChat development process
CONCLUSION
Chapter 6: Research findings
Chapter 7: Conclusion and further research
Chapter 8: Summary

Figure 1. The structure of this thesis

The thesis includes eight chapters. Chapter 1: Introduction gives a short introduction to this thesis, presents the thesis research background, motivations of this study and the research question. Chapter 2: Research design introduces the research approach, the applied research methods, and introduces how research data will be collected and analyzed. Chapter 3: Theoretical background covers the theoretical background of agile development, Scrum, test-driven development and other theories needed in this thesis. Chapter 4 and Chapter 5 demonstrate the development methods and the implementations of EasyScrum application and EasyChat application. Chapter 6: Research findings compare the two applications, EasyScrum and EasyChat, and introduces the main benefits TDD can bring to a software application development process. Chapter 7: Conclusion and further research concludes this thesis and suggests ideas for further study. Chapter 8 summarize the main findings of this thesis.

2 RESEARCH DESIGN

This chapter introduces the chosen research approach, the research chosen methods and shows how data will be collected and how it will be analyzed.

2.1 Research approach

In research design, there are various approaches that can be taken. There are three main research approaches: deductive, inductive, abductive. Deductive reasoning is the process of reasoning from theories to reach a logically certain conclusion. Inductive reasoning, in contrast, starts from collecting data and specific observation to generate or infer broad general ideas and theories. Different from deductive and inductive reasoning, abductive reasoning is based on incomplete data and aims to create or modify an existing theory, and additional data can be tested afterward. (Saunders et al. 2012 144-145.)

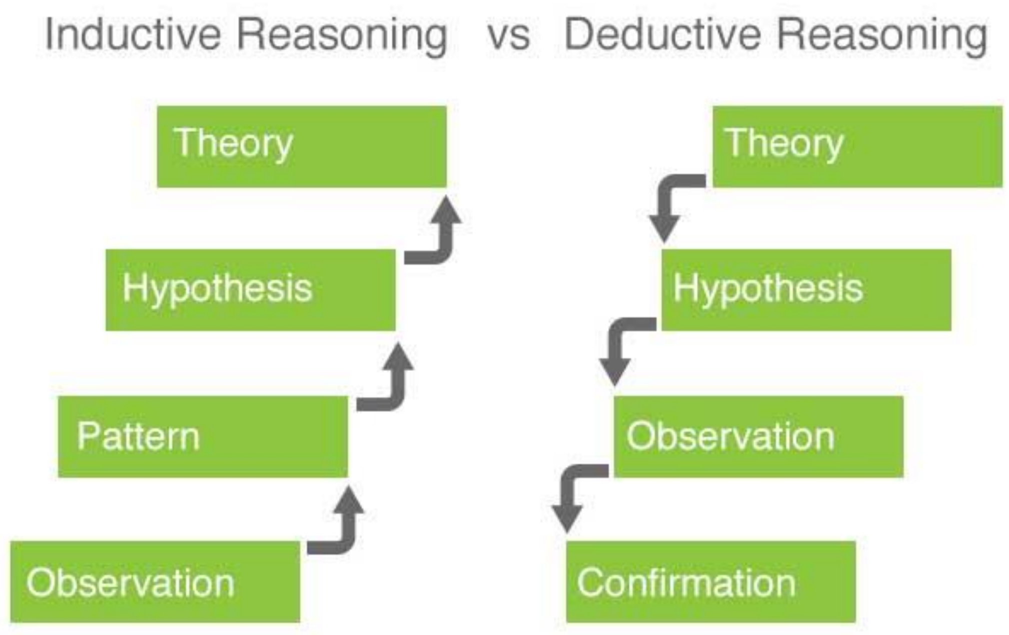


Figure 2. Inductive Reasoning vs. Deductive Reasoning (Elmansy, 2018)

This thesis studies an existing solution and applies it to the development process of a case study and then compares this to the existing case study. Moreover, the author analyzes data from the observation and experience to answer the research question. For the case of this thesis, with all those factors taken into account, deductive approach is adopted.

2.2 Research methods

Design science is a research method which mainly focuses on problem-solving. The primary purpose of design science is seeking an improvement in specific practices, techniques or technical capabilities to see which implementation can be more effective and efficient. The process of design science includes identifying problems, defining objectives of the solution, designing and developing, demonstrating and then providing the study. (Hevner & Chatterjee 2010.) This thesis applies a design science research method.

Quantitative research and qualitative research are the basic research approaches that are commonly used in research. Quantitative research is based on numerical and other measurable data to formulate and explain a particular phenomenon (Punch 2013, 3). Qualitative research, on the other hand, put the emphasis on understanding, and it might use non-numeric data such as images, videos, texts, recordings, etc. (Saunders et al. 2012, 161).

In research design, it is common to apply either qualitative research or quantitative research methods. Mixed-methods, which is the combination of qualitative and quantitative research designs, is becoming more and more common (Harwell 2011, 151). The author decided to apply qualitative research methods

2.3 Data collection and data analysis

Ethnography, grounded theory, and phenomenology are common methods used in qualitative research. Ethnography involves observation methods; the researchers are the direct observers and participants in their work environment over time. Moreover, there are different data collection methods such as interview, focus groups or action research. (Austin 2014.) Based on the case study presented here, the comparison of the development process of two applications, the author decided to use observation as a research method.

In participatory observation, keeping field notes is the primary data collection method. Field notes include the memos, the record of behaviors, events, and activities happened during observations. Field notes assist the researcher in gathering information to analyze and then answer the research questions. (Kawulich 2005)

The data are collected from different sources: case studies, books, articles and other sources. During the development process of the EasyScrum application, the author collects all the notes and reports to explain the limitations of the project. In order to achieve the objective of the thesis, the author gathers data by taking notes during EasyChat's development process. In addition to Scrum, test-driven development was applied in this process. The author then compares the two development processes to explain how TDD can improve a Scrum project. The development process and data analysis will be presented in Chapter 4 and 5, and the study will be discussed in Chapter 6.

3 THEORETICAL FRAMEWORK

To understand development methodologies and tools used in comparing the two applications, it is important to define the key development methodologies.

3.1 Methodologies

This subchapter discusses agile development methodology, Scrum, and test-driven development.

3.1.1 Agile development methodology

In the history of software development, the traditional approach to software development is the Waterfall methodology which is a sequence of development phases. In Waterfall methodology, each phase should be done before the new phase starts. In the 1990s, it was considered that software development should become more iterative and the agile development began to gain popularity. (Ashmore & Runyan 2014, 2-9.)

Agile is an umbrella term which includes several iterative lightweight approaches to software development and management. Agile software development focuses on communication, quality, and collaboration and it involves an iteration cycle with continuous feedback, continuous testing, and continuous integration. (Ashmore & Runyan. 2014, 2-9.) Extreme Programming (XP), Scrum, Crystal, Dynamic Systems Development Method (DSDM), Lean Development, and Feature-Driven Development (FDD) are the most popular agile methodologies (VersionOne 2018 2-10).

Several experienced software professionals convened in February 2001 to discuss the need for an alternative to the heavy process of Waterfall development. They formed the agile alliance, and after the conference, the principles and values were noted and are now known as The Agile Manifesto. (Ashmore & Runyan. 2014, 2-9.) The followings are the core values of the Agile Manifesto:

- Individuals and interactions over processes and tools
- Working with software over comprehensive documentation 9
- Customer collaboration over contract negotiation
- Responding to change over following plan

In addition to these values, the "Twelve Principles of Agile Software" were declared:

- Our highest priority is to satisfy the customer through the early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development. Agile processes harness change for the customer’s competitive advantage.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Business people and developers must work together daily throughout the project.
- Build projects around motivated individuals. Give them the environment and support they need and trust them to get the job done.
- The most efficient and effective method of conveying information to and within a development team is a face-to-face conversation.
- Working software is the primary measure of progress.
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- Continuous attention to technical excellence and good design enhances agility.
- Simplicity — the art of maximizing the amount of work not done — is essential.
- The best architectures, requirements, and designs emerge from self-organizing teams.
- At regular intervals, the team reflects on how to become more efficient, then tunes and adjusts its behavior accordingly.

(Ashmore & Runyan 2014, 7.)

3.1.2 Scrum

Scrum is one of the most popular agile development frameworks. The term Scrum first appeared in an article in the January 1986 Harvard Business Review, entitled "The New Product Development Game." Then, in 1993, Jeff Sutherland developed Scrum development methodology. Scrum is an agile framework for collaborations, project control, requirements adaptation and continual deliverable of highest quality products to the client every 2-4 weeks. (Ashmore & Runyan 2014, 54.)

Benefits of implementing Scrum

According to VersionOne, Scrum brings these benefits for organizations that adopted Scrum.

- Better quality
- Decreased time to market
- Higher customer satisfaction
- Increased collaboration, project control, and ownership
- Higher team dynamics
- Reduce risks

(VersionOne 2018.)

Components of Scrum development methodology

- Product owner, the voice of the customer, responsible for managing requirements, optimizing and ensuring the development team understand the value of work.
- Scrum master, the supporter leader of the team, manages Scrum process, ensure team communication, and coach teams and organizations to understand, follow the rules and practices. Scrum master can be technical lead or lead developer of the group or project manager.

- The development team is responsible for developing functionality; they are accountable for planning how to put the product backlog into potentially shippable increments features. The development team is cross-functional, every member of the team can be responsible for different roles, for example, a member can be designer, developer or tester.

(Resnick & Bjork 2011.)

Scrum process

The Scrum software development methodology consists of sprint, sprint planning, daily scrum, sprint review, and sprint retrospective. All events in Scrum are time-boxed. Also, in Scrum events, the team uses these following artifacts: Product Backlog, Sprint Backlog, and Increment to maximize the transparency and help other members of the team have the same understanding.

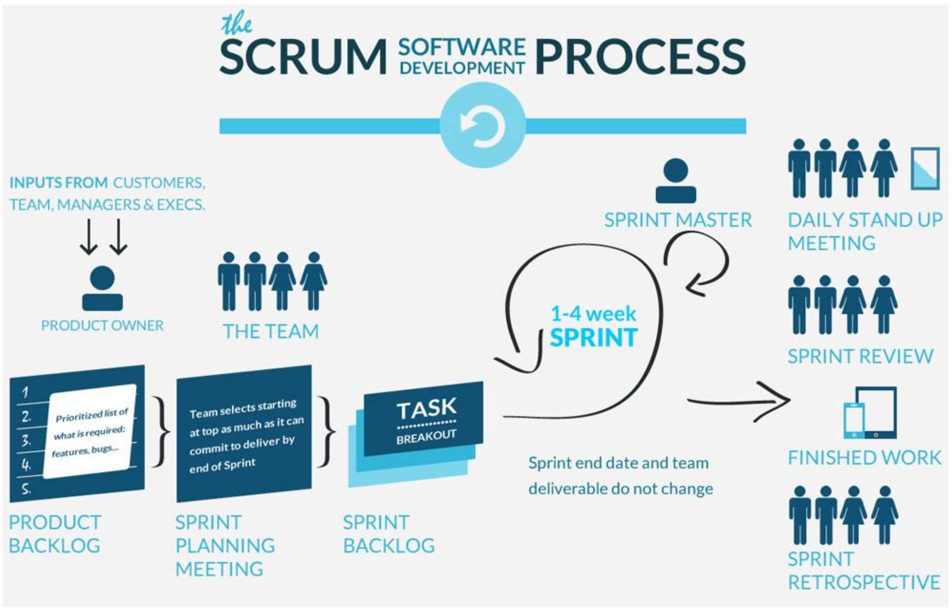

the SCRUM SOFTWARE DEVELOPMENT PROCESS
INPUTS FROM CUSTOMERS, TEAM, MANAGERS & EXECS.
PRODUCT OWNER
THE TEAM
SPRINT MASTER
DAILY STAND UP MEETING
1-4 week SPRINT
SPRINT REVIEW
Prioritized list of what is required: features, bugs...
Team selects starting at top as much as it can commit to deliver by end of Sprint
TASK BREAKOUT
Sprint end date and team deliverable do not change
FINISHED WORK
PRODUCT BACKLOG
SPRINT PLANNING MEETING
SPRINT BACKLOG
SPRINT RETROSPECTIVE

Figure 3. Scrum software development process (Soares 2016)

Scrum starts with Scrum planning meeting; sprint planning focuses on two questions: “What should be and can be delivered in the sprint Increment?”

and “How will the work needed for the execution of sprint be achieved?”. During the meeting, the scrum team discusses the functionalities to be developed in the product and create product backlog items. The Scrum team will estimate and select the prioritized specific product backlog items to deliver after each sprint. The team, then, identifies the sprint goal which is a description of what the team planned to achieve during the sprint and pulled the product backlog items into the sprint Backlog which can be a tracking system, project management software or a whiteboard, etc. The Sprint Backlog consists of these Increments: “product backlog items” list, “to-do” list, “in progress” list and “done” list (Pham & Pham 2012, 6-14.)

Scrum sprints are in a time-boxed of 2-4 weeks, but it can be shortened to one week. During the sprint, there will be Daily Scrum, a 15-minute meeting, where each member of the team explains what he did, what he will do and are there any problems that he encountered. Then the team can create a plan for the next day in the sprint. Throughout the sprint, the team members are responsible for modifying the Sprint Backlog between the Increments (Pham & Pham 2012, 6-14.)

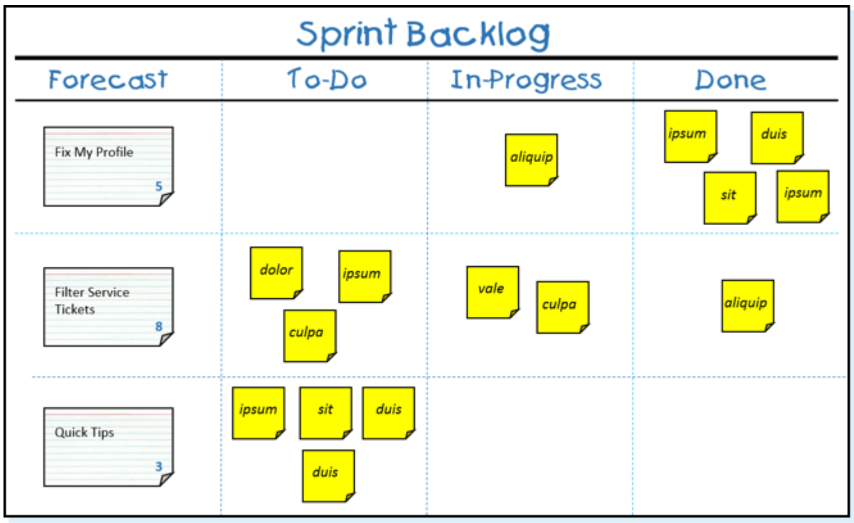


Figure 4. Example of Sprint Backlog (Scrum 2018)

At the end of the sprint, a group meeting will be held between every component including Product owner, the key Stakeholders, and the Scrum team. During the sprint review, the Scrum team and Stakeholders discuss what was done and what was not done, the product owner and the stakeholders will give feedback and either accept or reject the work. Sprint retrospective is the final team meeting to discuss what went well, what went unexpectedly and how the team can improve in the next sprint. The benefits of the retrospective are helping the Scrum team to improve and make the next sprint more effective (Pham & Pham 2012, 6-14.)

3.1.3 Test-Driven Development

Test-driven development is a software development advanced technique of writing an automated unit test for a specifically required functionality before writing any code. By using this technique, the developers can have the feedback at any time that the software is still working and avoid duplication of code. Different from the traditional testing method that developers usually write test and test after every implementation code is written, Test-Driven technique starts with the developers develop and write the test first before the actual development of the application. This force the developers thinking about the requirement, the code design and the use of methods before implementing it, so that it makes sure the developer implements what is required. (Bender & McWherter 2011, 8-10.)

The test-driven development cycles

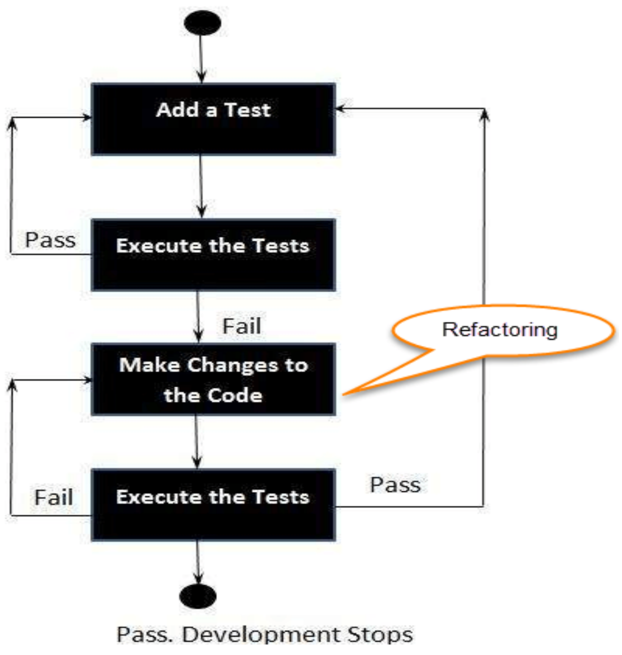


Figure 5. Test-driven development steps (guru99 2018)

There are three steps in TDD process:

Step One: Write a failing test

The concept of test-driven development is first to understand the requirements and features need to be done and create the test, the developers write just enough code for compiling, run the test and expect it to fail. The test will fail because the actual functionality has not been developed.

Step two: Make the test pass

In this step, write enough production code to make the test pass. The developers can execute the automated test at any time to make sure their code is working, and they can have immediate feedback.

Step three: Refactor

After the test pass, the developers can change the code and remove duplication without changing the behavior if necessary to improve the structure and the design of code. The developers should re-run the test to ensure all the test passed.

Step four: Repeat

As the test pass, the developers can repeat all the steps to develop another functionality.

(Palermo 2006.)

Benefits of test-driven development

When adopting test-driven development technique, the developers will be more productive while coding. In the development process, the developers write one failing test and focus on finding the solution for a specific functionality rather than thinking about every feature and the whole application at a time (Winter 2016). Also, according to Jeffrey Palermo, test-driven development forces the developers to imagine how the functionalities will work and the development team must fully understand the requirements to write the test and the production code. Moreover, by using TDD methodology, the development team will have immediate feedback on each component that the team is developing and testing (Palermo 2006).

In the TDD process, the developers only need to write enough production code to make the test pass; this will help the code easy to be maintained, flexible to be fixed and secure to be extended. Since testing in TDD is integrated into the development process; when the developer makes a change, all the other existing tests will be rerun to ensure the code is still working, this also helps to decrease the chance of having several bugs and save time spent on finding and fixing bugs. When a test pass, the developers can change the code and remove duplication without breaking the behavior if necessary to attain clean design and clean code. (Bender & McWherter 2011, 8-10.)

Overall, with continuous feedbacks, clean and working code, test-driven development increase and improve the software quality and help to save time.

3.2 Technologies and tools

3.2.1 Git & GitHub

Version Control Systems (VCS) is a software that helps teams and organizations to manage changes in every file in the project over time and to have the capability of coming back to a specific version of the application whenever needed. There are different types of VCS: Local Version Control system (LVCS), Centralized Version Control System (CVCS), and Distributed Version Control System (DVCS) (Somasundaram 2013, 8-11.)

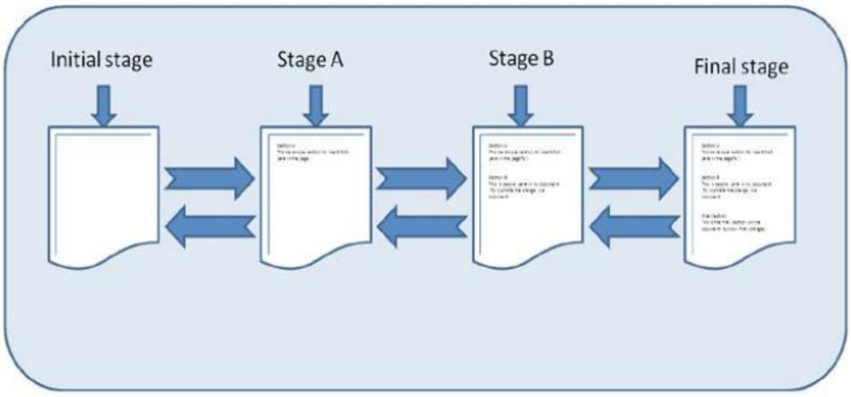


Figure 6. The flow of content with Version Control System (Somasundaram 2013.)

There are many different workflows of Version Control Systems that the developer teams can choose to make use of VCS. One of the benefits of VCS is enhancing collaboration, this help to prevent conflicts during the work. Everybody in the team can work independently with any files at any time without overwriting others work. Also, Version Control Systems stores different versions of the project which the developer can re-open for checking or restore the previous version. Moreover, different changes in files required the description of what changes were made so that the developer can understand what happened (Atlassian 2018.)

Git

Git, an open source Distributed Version Control System, was developed by Linus Torvalds in 2005. Git is one of the most popular VCS and is famous for its high-performance, flexibility, and security. According to Atlassian, every performance characteristics of Git such as committing, branching, merging and comparing are optimized to attain high-performance. By using SHA-1, a cryptographically secure hashing algorithm, Git files and content are secured and protected in case of accidental and harmful change, and Git history is possible to track down. Git supports different development workflow, both small and large project, and compatible with many different systems and protocols. (Somasundaram 2013, 15.)

GitHub

GitHub is a platform for collaborative development. It is a web-based service which hosts open source and private team project using Git. GitHub has become influential in the open source development community with more than 79 million projects hosted on GitHub (GitHub 2018.)

3.2.2 WebSocket and Socket.io

WebSocket is a technology that supports two-way communication between client and server using a TCP socket to create an efficient connection. Although it is designed to be used for web applications, programmers can still include them in any application (Rai 2013.)

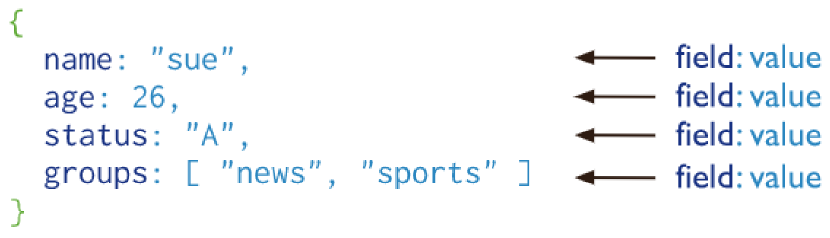
Socket.io, a WebSocket API, is a JavaScript library for making real-time web applications and mobile application. With its powerful and easy-to-use interface, Socket.IO is becoming more and more popular with developers and companies such as Microsoft Office, Yammer, Trello, hackathons, and start-ups (Rai 2013.)

3.2.3 MERN stack

MERN is an open source combo consists of four most popular JavaScript related technologies: MongoDB, Express, ReactJS, and Node.js. Developers use the MERN stack to build a Universal app. (Morgan 2017)

MongoDB

MongoDB is an open source NoSQL database which is non-SQL, non-relational databases. NoSQL includes many types of databases such as Document Database, Columnar Database, and Graph Database (AWS 2018). MongoDB, the most famous NoSQL database, is a document database which saves data as JSON documents (Krol 2014). The following figure shows what a JSON object (document) look like in MongoDB.



```
{
   name: "sue",
   age: 26,
   status: "A",
   groups: [ "news", "sports" ]
}
```

Figure 7. MongoDB JSON object (MongoDB 2018)

According to MongoDB, the advantage of using MongoDB are:

- Flexible and dynamic schema design
- Capable of fast querying and indexing
- High-performance, high-availability and easy to scale

(MongoDB 2018.)

Express

Express is a framework for NodeJS. Express includes many powerful features on the web platform as well as on mobile applications. Express supports HTTP and middleware methods to create an extremely powerful and easy-to-use API. Some of the main functions of Express can be summarized as follows:

- Establish intermediate classes to return HTTP requests
- define router for the use of different actions based on HTTP method and URL
- Allows returning HTML pages based on parameters.

(Mozilla 2018.)

ReactJS

ReactJS is a JavaScript library for building user interfaces and is developed by the Facebook team and Frontend Developer often uses ReactJS in Web Application Design. React provides an easy way to build interfaces by developing components and combining them. It also provides a Virtual DOM method for optimizing rendering as well as allowing for rendering via Node.js. Also, it implements one-way reactive data flow, which keeps everything modular and fast (Reactjs 2018.)

NodeJS

NodeJS is a server-side framework/platform based on the JavaScript V8 Engine platform. It is an open source JavaScript cross-platform runtime environment for building server-side applications such as video clips, forums, and especially social networking sites (Nodejs 2018.)

3.2.4 Mocha

Mocha is an open source JavaScript testing framework running on NodeJS which allows asynchronous testing. The following snippet demonstrates the example of mocha test code.

```
var assert = require('assert');

describe('Array', function() {

  describe('#indexOf()', function() {

    it('should return -1 when the value is not present', function() {

      assert.equal([1,2,3].indexOf(4), -1);

    });

  });

});
```

Mocha has different features to support the process of testing and has many command lines to provide needed information in the testing report. For examples:

- Support simple asynchronous
- Provide before(), after(), beforeEach(), and afterEach() functions to support set up and clean up test environment.
- Provide “watch” function to re-run the test automatically every time the file changes
- Provide “diff” function to show the differences between the expected result and the actual result.
- Allowing to use many assertion libraries such as chai.js or expect.js.

(Mochajs 2018.)

4 CASE: EASYSCRUM APPLICATION DEVELOPMENT PROCESS

This chapter includes the project background and the development process of the EasyScrum application. By the end of this chapter, the reader can see the results of the application which only applied Scrum methodology.

4.1 Introduction

During three agile software development courses in fall 2017, the students had learned and applied agile development methodology to develop the EasyScrum application. EasyScrum is a Scrum toolkit that helps teams in their agile workflow and during the development process.

4.2 Project goals

The purposes of developing the EasyScrum application are to understand and practice agile software development methodology learned during agile courses in the study modules. Also, the students can practice creating a functional and high-quality application by applying knowledge and skills learned from one of the courses in agile modules which is Web Application Development.

4.3 The development team

The project implements and practices Scrum artifacts, therefore, the team consists of the Product Owner, Scrum Master, and the development team. The Scrum roles are arranged based on the skills of the team members and choice of the team member. In the team, the team members can hold more than one roles.

4.4 Project coordination

The team performed Scrum as a framework for collaboration and in Scrum, the team control and self-organizes the work. Therefore, the team decided each sprint last for two weeks to three weeks. Before the sprints start, the team organized planning meeting to evaluate each task and decide which tasks to do in each sprint and divided the tasks. The development process of the project includes three sprints.

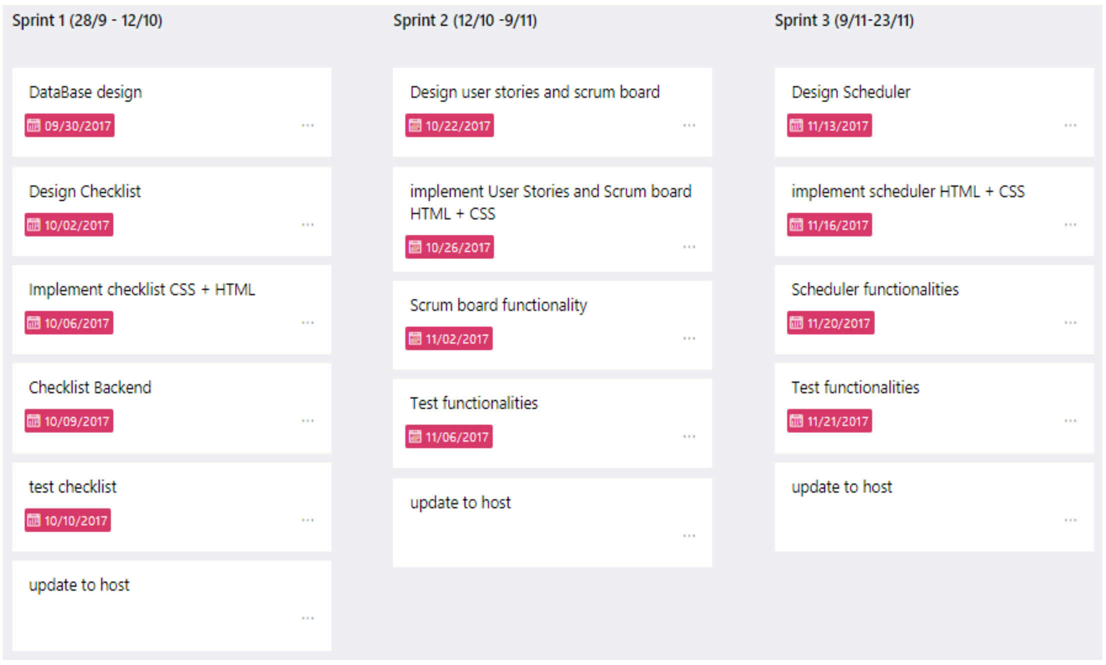
Throughout the sprint, the team had several check-up meetings for tracking and checking the working progress of each member. In this meeting, each member answered these questions:

- What did the member do?
- What will the member do today?
- Are there any impediments in the member’s way?

At the end of the sprint, the group had the retrospective meeting to discuss what the team had done and what the team had not done in the sprint and to demonstrate the work to the stakeholders. The purpose of the meeting is to receive feedback to improve the application and the working process.

4.5 Development process

Figure 8 demonstrates the summary of three sprint planning with the start and end date of the sprints. Also, each task in the sprints has a deadline.



Sprint 1 (28/9 - 12/10)

DataBase design
09/30/2017

Design Checklist
10/02/2017

Implement checklist CSS + HTML
10/06/2017

Checklist Backend
10/09/2017

test checklist
10/10/2017

update to host

Sprint 2 (12/10 -9/11)

Design user stories and scrum board
10/22/2017

implement User Stories and Scrum board HTML + CSS
10/26/2017

Scrum board functionality
11/02/2017

Test functionalities
11/06/2017

update to host

Sprint 3 (9/11-23/11)

Design Scheduler
11/13/2017

implement scheduler HTML + CSS
11/16/2017

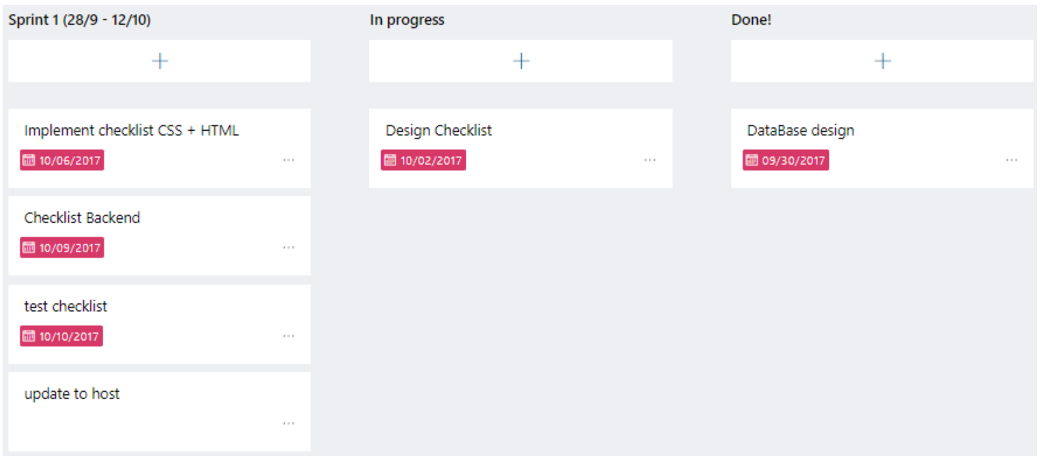
Scheduler functionalities
11/20/2017

Test functionalities
11/21/2017

update to host

Figure 8. EasyScrum sprint planning

The team used Microsoft Team for task management. After the sprint planning, all the tasks in each sprint went to the Sprint Backlog which includes these increments: “To-Do,” “In Progress,” and “Done.” The following figure introduces the sprint 1 of the EasyScrum application project.



Sprint 1 (28/9 - 12/10)

Implement checklist CSS + HTML
10/06/2017

Checklist Backend
10/09/2017

test checklist
10/10/2017

update to host

In progress

Design Checklist
10/02/2017

Done!

DataBase design
09/30/2017

Figure 9. Sprint 1 of the EasyScrum application project

At the beginning of the project, the team decided on the development flow that the team designs first, develops the functionalities and then tests in the end. For instance, in the first sprint, the team develop checklist feature including create, read, update and delete functions. After the development of functionalities completes, the developers test every function in the checklist feature, and the developers test manually. sprint 2 and sprint 3 applied the same development process in sprint 1, and because during the project and after the project, the team gathered notes and wrote reports of how the project went and what were the drawbacks and limitations. Therefore, the author considers not to explain the sprint 2 and sprint 3 in detail.

4.6 Project results

At the end of the course, the team was able to finish every feature and delivered the fully functional application to the stakeholders. Also, the stakeholders accepted and satisfied with what the team delivered. The project is now deployed to the hosting server and published to GitHub as an Open Source project.

4.7 Project drawbacks and limitations

Even though the team delivered a fully functional application and the stakeholders satisfied; the team had encountered several drawbacks and limitations during the development process.

Due to the reports, one of the problems is that the developers did not have enough time to test the functionalities before delivering the products to the stakeholders in each sprint. The team used GitHub as the version control and everyone on the team has a branch. However, some members had limited skills or never use GitHub before so that there were often conflicts and problems had not been fixing before merging into the master branch. Therefore, the team ended up spending more time on resolving conflicts. Because of this reason and because the tests are done after the development, the team usually did not have enough time to test and fix every function before the sprint ends.

Besides, the team did the testing part manually due to the lack of testing knowledge instead of testing automatically using the testing frameworks which can save more time and more efficient. Therefore, it costs more time for the team in testing also. In the sprint retrospective, the team usually delivered the application with some functions that were not working, and the team had to delay the next sprint and extended the deadline for testing.

5 CASE: EASYCHAT APPLICATION DEVELOPMENT PROCESS

This chapter includes the project background and the development process of the EasyChat application. By the end of this chapter, the reader can see the results of the application and the benefits of applying test-driven development.

5.1 Project introduction

EasyChat is a real-time chat web-based application which was developed by the author in January 2018. The target audience of the application is for students, teams or groups who need an application to communicate faster and easier. Moreover, EasyChat aims to point out the benefits of applying Testing-Driven Development along with Scrum.

5.2 Project goals

The EasyChat application aims to create a real-time chat application that supports teams, students, and group communication and collaboration. Moreover, the author creates the application to apply Scrum methodology and test-driven development technique. The main goal of this application is to identify the benefits of applying test-driven development to support Scrum.

5.3 Project specification and the development team

Since the developer came up with the idea to create an application to apply test-driven development in Scrum, the developer is responsible for all roles including Scrum Master, Product Owner and Stakeholders and the development team. The roles in the development team include designer, front-end developer, back-end developer and tester.

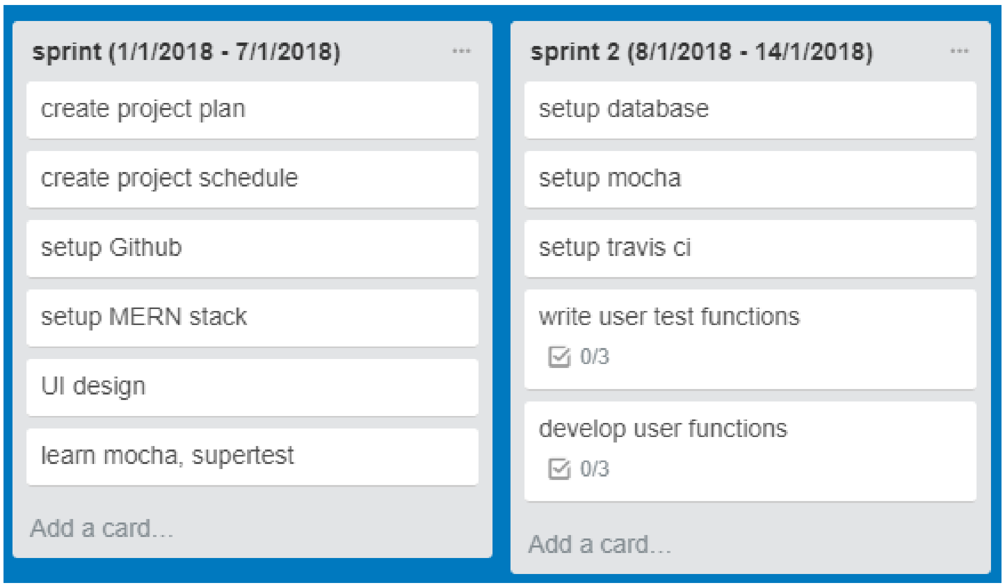
The EasyChat application is a chat application, and it includes essential features that a chat application could have, for example, create room, delete room or invite others to the conversation. As mentioned above, there is only one person in the development team so that the developer only develops and deploy the project to localhost and host the project on GitHub for the automated testing purpose. Also, the developer uses Git and GitHub to manage every version of the application.

5.4 Project coordination

The project also applies Scrum methodology. Therefore, the project has the same management as the EasyScrum application. However, there are some differences that the development team has only one person, due to this reason, the use of communication tool is not necessary. The developer will manage every task in Trello, Trello is a web-based project management application which supports keeping track of everything. The EasyChat project includes four sprints in total.

5.5 Development phases and Schedule

Figure 10 and figure 11 demonstrate four sprints in the EasyChat application project. Each sprint in the project have a start date and end date and includes different tasks to do.



sprint (1/1/2018 - 7/1/2018)

- create project plan
- create project schedule
- setup Github
- setup MERN stack
- UI design
- learn mocha, supertest

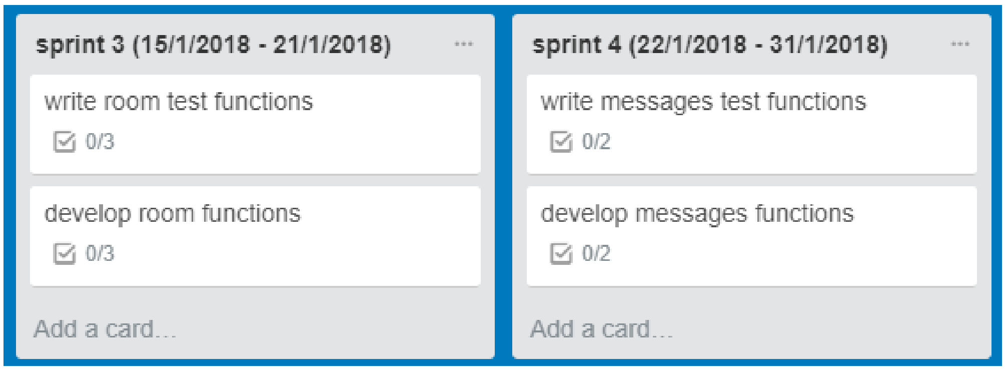
Add a card...

sprint 2 (8/1/2018 - 14/1/2018)

- setup database
- setup mocha
- setup travis ci
- write user test functions 0/3
- develop user functions 0/3

Add a card...

Figure 10. EasyChat sprint planning



sprint 3 (15/1/2018 - 21/1/2018)

write room test functions
0/3

develop room functions
0/3

Add a card...

sprint 4 (22/1/2018 - 31/1/2018)

write messages test functions
0/2

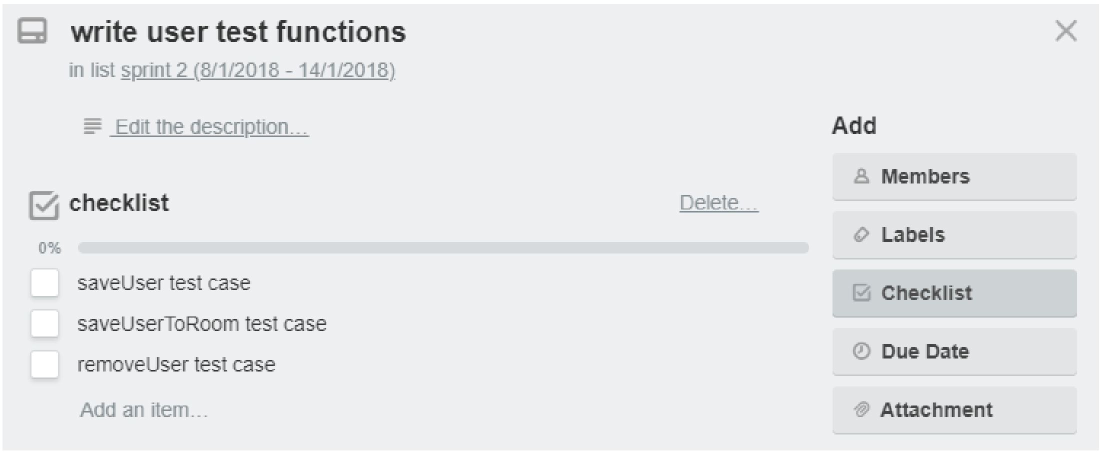
develop messages functions
0/2

Add a card...

Figure 11. EasyChat sprint planning

The first sprint in the project consists of setting up and learning tasks. The tasks are to create plan, schedules and to set up the project base. In the next sprint, the developer continues to setup database and testing framework, then write test and develop the first function. The next two sprint includes developing other needed functions in the application and finishing up steps. The project management of this project is the same as EasyScrum application, each sprint in EasyChat project also has different increments including “To-do,” “In Progress,” and “Done.”

In each development task, there is a checklist for different requirement of the application. The below figure shows the “write user test functions” checklist.



write user test functions

in list sprint 2 (8/1/2018 - 14/1/2018)

Edit the description...

checklist

Delete...

0%

- saveUser test case
- saveUserToRoom test case
- removeUser test case

Add an item...

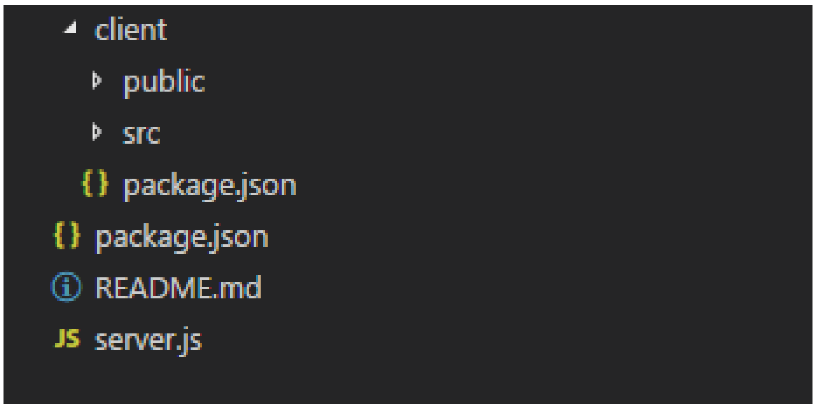
Figure 12. Development task

5.6 Development process

Install and set up MERN application base

EasyChat is a chat application that allows users to create chat rooms and send messages. Therefore, setting up a server and database is necessary for storing data and for sending the requests from client-side which is the application itself, to the server to retrieve data from the database. Moreover, the developer aims to test the database functions consists of “create,” “read,” “update,” and “delete” functions and all the automated test for database functions are in the server-side. The author planned to use Express which is a NodeJS framework for running the server. Also, ReactJS is used to implement the application which is client-side. Therefore, the author chose to use MERN stack which has client-side and the server-side.

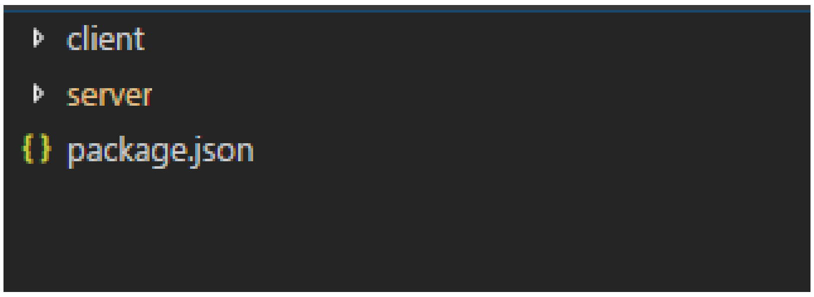
The developer uses the BradTraversy’s React-Express start pack which is an open-source on GitHub to start the project. The starter pack includes client-side and server-side as shown below.



```
client
  public
  src
  package.json
package.json
README.md
server.js
```

Figure 13. react-express start pack structure

The developer, then, re-organized the structure of the project so that there are two folders: client and server.

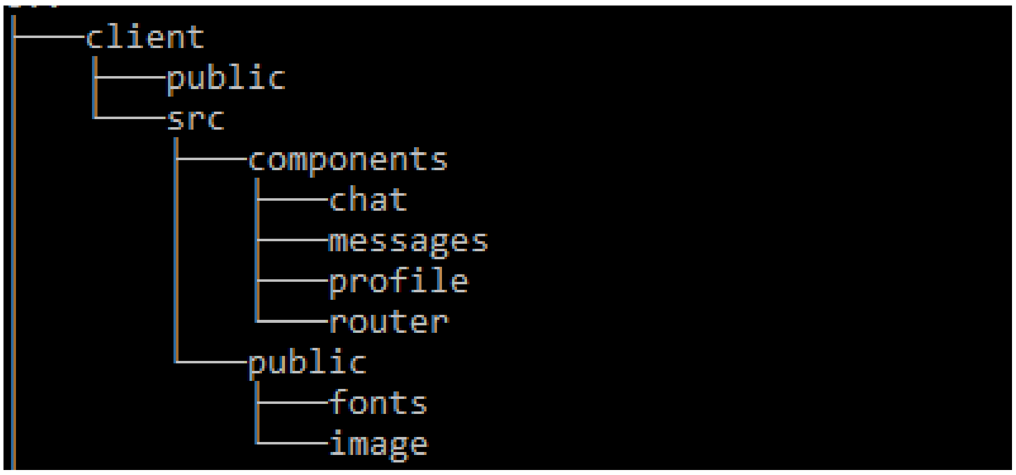


```
client
server
{} package.json
```

Figure 14. EasyChat structure

The client folder includes different components of the application which are the user interfaces. The chat component is for showing the chat room, chat room list and the chat window, the messages component is for showing the messages in the conversations and the profile is for showing the user profile. In specific component, there are different functions for communicating to the server and controlling the user interfaces.

The following figure shows the client folder structure.

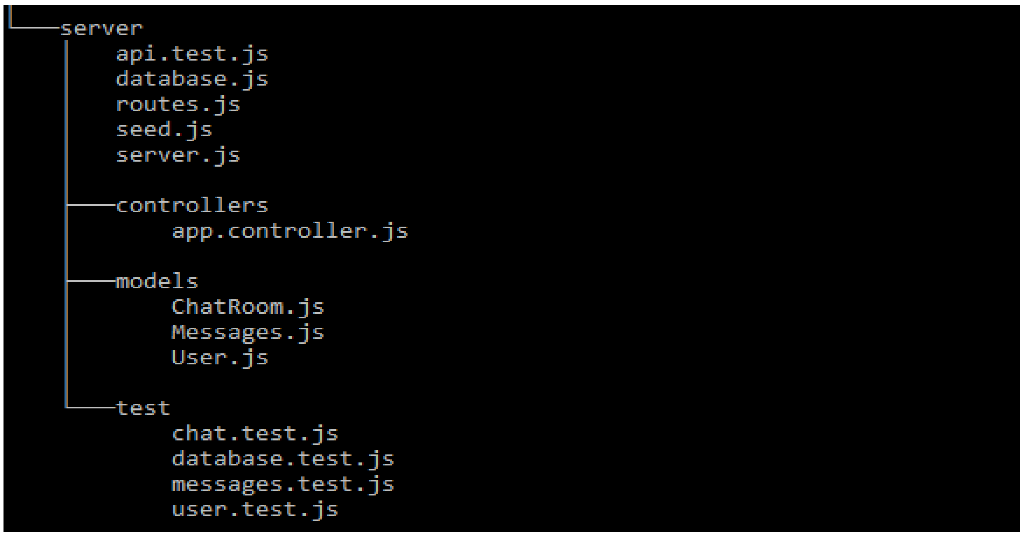


```
├───client
    ├───public
    └───src
        ├───components
        │   ├───chat
        │   ├───messages
        │   ├───profile
        │   └───router
        └───public
            ├───fonts
            └───image
```

Figure 15. Client folder structure

The server folder consists of the database, routes and the server files. Also, there is a “controller” file to control the database functions and also the test folder which includes all the test files.

The following figure shows the server folder structure.



```
└───server
    │   api.test.js
    │   database.js
    │   routes.js
    │   seed.js
    │   server.js
    │
    ├───controllers
    │       app.controller.js
    │
    ├───models
    │       ChatRoom.js
    │       Messages.js
    │       User.js
    │
    └───test
            chat.test.js
            database.test.js
            messages.test.js
            user.test.js
```

Figure 16. Server folder structure

All dependencies need to be installed in both client-side and server-side to run the application. The following command lines show how the dependencies can be installed according to Quick Start of the React-Express starter pack.

Install dependencies for server

```
$ npm install
```

Install dependencies for client

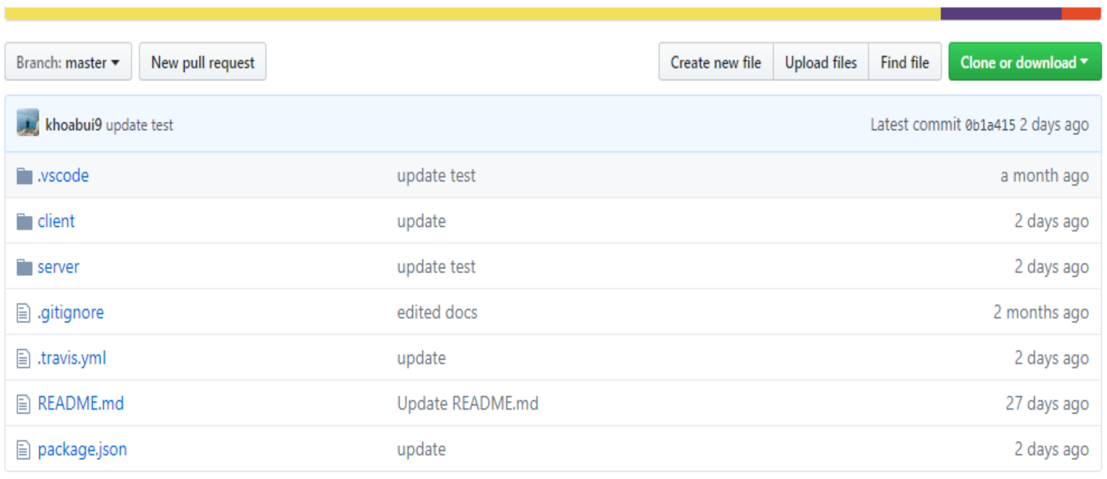
```
$ npm run client-install
```

Set up Git and GitHub repository

After creating the project foundation, Git is initialized to keep track of the files and the versions in localhost. Also, a GitHub repository is created for hosting the files and project versions online and for the automated testing purpose which will be explained later in this chapter. The following command lines demonstrate how the developer created initialize Git and committed the files to GitHub.

```
$ git init

$ git add --all

$ git commit -m "first commit"

$ git remote add origin
https://github.com/khoabui9/chat.git

$ git push origin master
```

The following figure shows the EasyChat GitHub repository.



khoabui9 update test — Latest commit 0b1a415 2 days ago

.vscode	update test	a month ago
client	update	2 days ago
server	update test	2 days ago
.gitignore	edited docs	2 months ago
.travis.yml	update	2 days ago
README.md	Update README.md	27 days ago
package.json	update	2 days ago

Figure 17. EasyChat GitHub repository

Install and setup testing environment

For implementing test-driven development, Mocha testing framework and Chai library need to be installed. Chai is a behavior-driven development/test-driven development assertion library for node and browser which can be paired with any JavaScript testing framework. The following commands are executed to install Mocha and Chai.

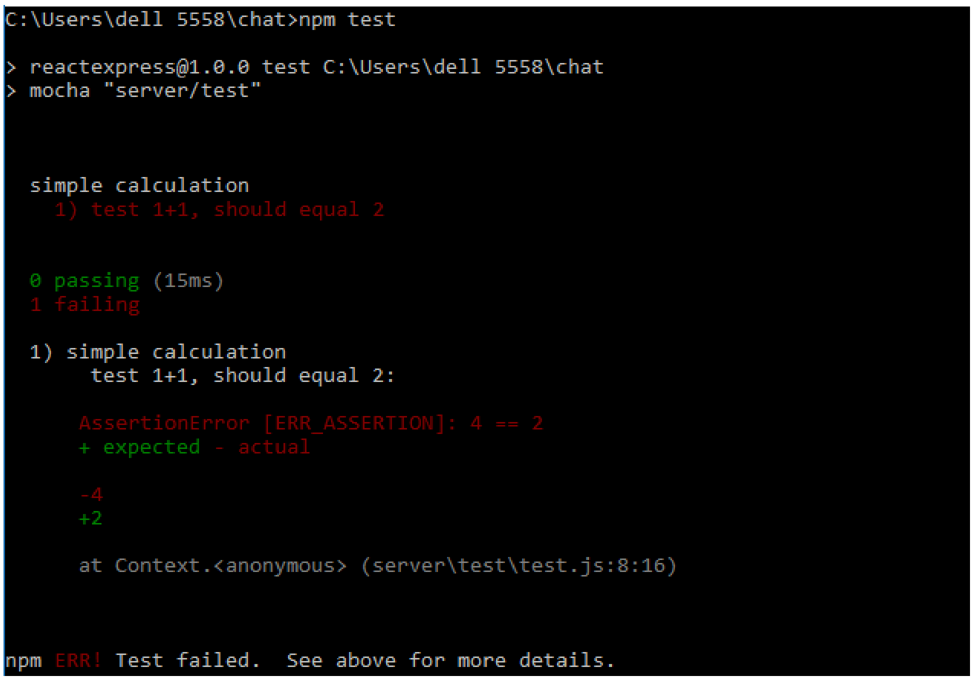
```
$ npm install -g mocha

$ npm install –save-dev chai
```

After the application have Mocha and Chai installed, All the test files are put under a directory called "test." Because the test folder is under the server directory, a path to the test folder was put in the configuration file, so that the developer can execute the test command without changing the directory to the server folder. After the installation and configuration, a simple test case is created to make sure that the test framework works as expected.

```
describe('simple calculation', function() {
    it('test 1+1, should equal 2', function() {
        assert.equal(4,1+1);
    });
});
```

The above test case code tests the simple equation 1 + 1 equals 4 since 1 + 1 equals 2, the test will fail. After writing the sample test case, the following command is executed to run the test.

```
$ npm test
```



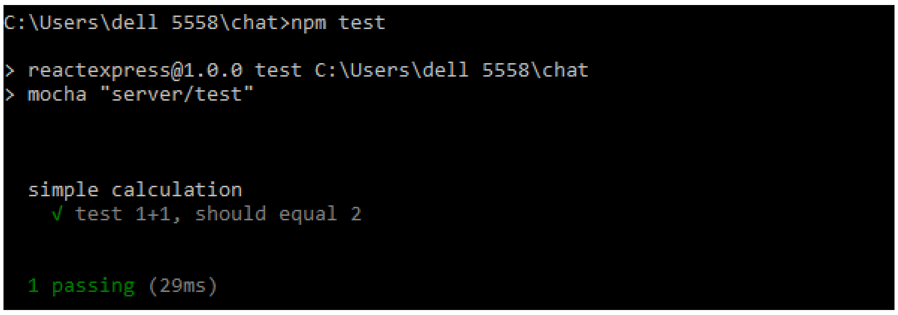
```
C:\Users\dell 5558\chat>npm test

> reactexpress@1.0.0 test C:\Users\dell 5558\chat
> mocha "server/test"



  simple calculation
    1) test 1+1, should equal 2


  0 passing (15ms)
  1 failing

  1) simple calculation
       test 1+1, should equal 2:

      AssertionError [ERR_ASSERTION]: 4 == 2
      + expected - actual

      -4
      +2

      at Context.<anonymous> (server\test\test.js:8:16)



npm ERR! Test failed.  See above for more details.
```

Figure 18. Failed test case result

After running the test, the testing report as shown in the above figure, demonstrate that the test has failed. The expected result which is the green color should equal “2,” and the actual result is “4” which is in red. The test cases, then, has been changed to make the test pass as follows.

```
assert.equal(2,1+1);
```

the following figure shows the successful test case.

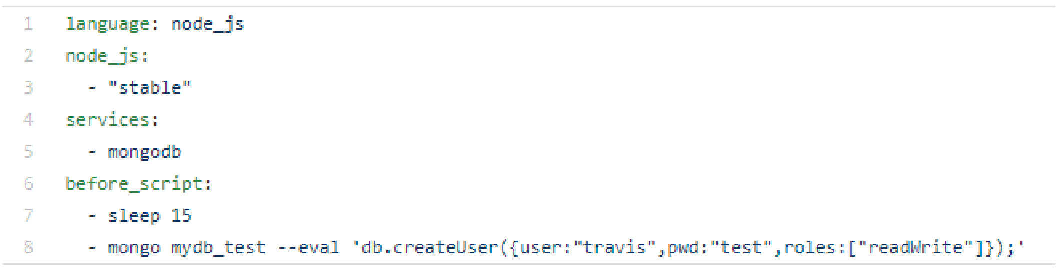


```
C:\Users\dell 5558\chat>npm test

> reactexpress@1.0.0 test C:\Users\dell 5558\chat
> mocha "server/test"



  simple calculation
    √ test 1+1, should equal 2


  1 passing (29ms)
```

Figure 19. Successful test case result.

The application uses MongoDB database to store the data and every run or run the test related to the database; another command prompt needs to be opened to start the MongoDB database. Also, the author uploads the project to GitHub for Open-Source purpose and wants the working version is on GitHub. Therefore, the developer decided to use Travis-CI, which is a hosted, continuous integration for automated testing public projects hosted project at GitHub.

For setting up the Travis-CI service, it is necessary to navigate to the GitHub repository, then, add the third-party service Travis-Ci and allow it perform actions when events occur in the repository. Besides, a file named “.travis.yml” needs to be added to the projects and configured as below for Travis-CI to recognize and run.



```
language: node_js
node_js:
  - "stable"
services:
  - mongodb
before_script:
  - sleep 15
  - mongo mydb_test --eval 'db.createUser({user:"travis",pwd:"test",roles:["readWrite"]});'
```

Figure 20. Travis-CI configuration

Development workflow

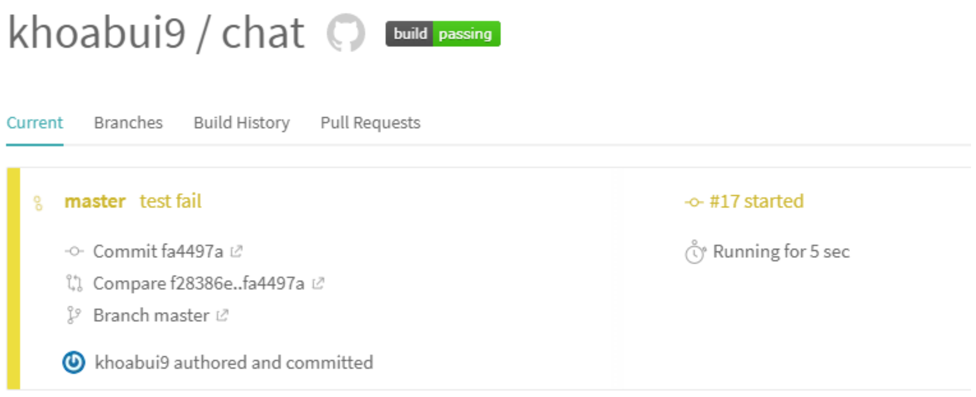
The development applied test-driven development software techniques along with Scrum, as mentioned above in theoretical framework section, the first step of TDD is writing the test first. When the user enters the web application, the first page to see is the login page which will need the function that adds the user to the database. The “saveUser” function is the chosen function to demonstrate TDD. The test case for this function is to see whether it save the user to the database or not. Writing the test first will force the developer to think what should be tested and what should be in the actual function. In this case, to save the user to the database, it is necessary to create a User object which should have the name property.

```
describe('save and find Record', function () {
  before(function (done) {
    var name = "testing user";
    controller.saveUser(name);
  });
it('Should return data saved to the database', function
(done) {
  User.findOne({name: "testing user"}, (err, data) => {
      data.should.have.property("name");
      data.name.should.equal("testing user");
      done();
    });
  });
});
```

In the code snippet above, first, it calls the “saveUser” function with the parameter which is the name of the user and then calls MongoDB “findOne” function to see if it is saved to the database. After finishing writing the test, the application will be committed to GitHub to run the test in Travis-CI service to check if the test fails. The test will fail since the function “saveUser” has not been developed yet.

```
$ git add --all

$ git commit -m “test fail”

$ git push origin master
```

After pushing to the master branch of the project on GitHub, Travis-CI build and run the Mocha test automatically.


khoabui9 / chat build passing

Current Branches Build History Pull Requests

master test fail

Commit fa4497a

Compare f28386e..fa4497a

Branch master

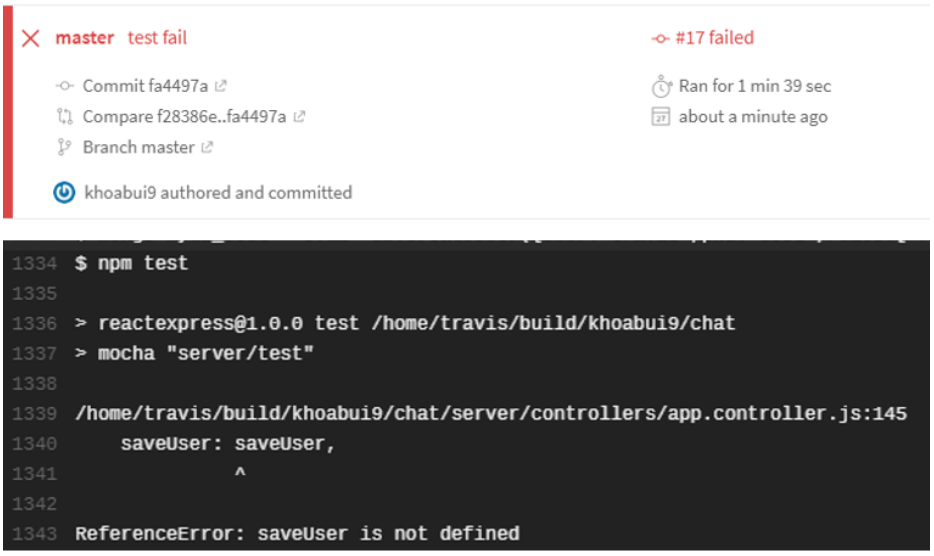
khoabui9 authored and committed

#17 started

Running for 5 sec

Figure 21. Test is running in Travis-CI

After running for more than one minute, the test marked as fail. The following figure shows that the test failed, time ran and on which branch of the project, and it also provides the report that the “saveUser” function is not defined.



master test fail

Commit fa4497a

Compare f28386e..fa4497a

Branch master

khoabui9 authored and committed

#17 failed

Ran for 1 min 39 sec

about a minute ago

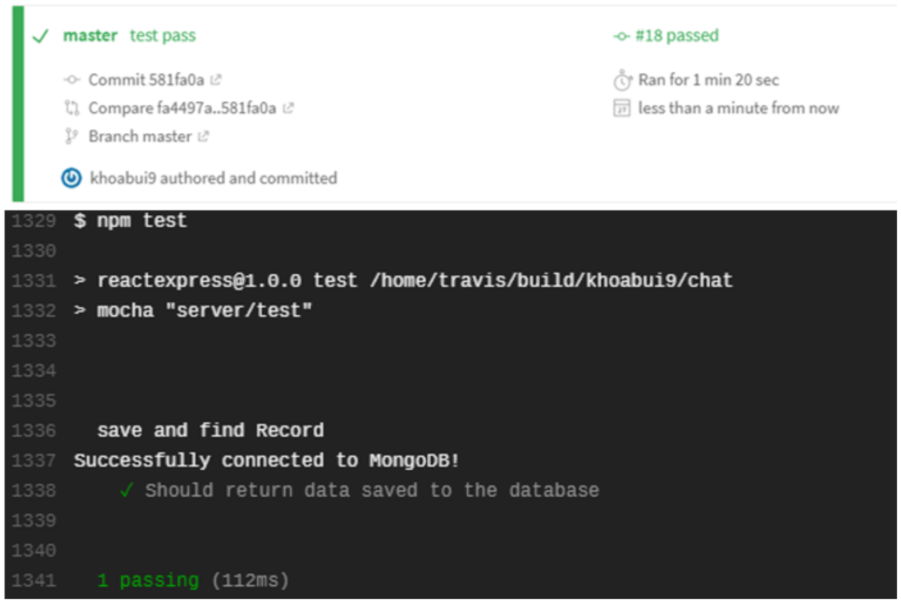
```
$ npm test

> reactexpress@1.0.0 test /home/travis/build/khoabui9/chat
> mocha "server/test"

/home/travis/build/khoabui9/chat/server/controllers/app.controller.js:145
    saveUser: saveUser,
              ^

ReferenceError: saveUser is not defined
```

Figure 22. Failed test report

The next step when applying test-driven development technique is starting to develop the function with just enough production code. The “saveUser” function, then, is developed and pushed to GitHub again to see if the test pass. This step will base on the test case to develop the actual function. In the function, the property name of the new user will be the parameter. The following code snippet demonstrates the “saveUser” function.

```
saveUser = (name) => {
    var user = new User({
        name: name
    });
    user.save(function (err,data) {
        if (err) return err;
    })
}
```

After finishing the function, the developer commits the changes and push to GitHub. After pushing to GitHub, the Travis-CI build and rerun the test. The following figure is the result of the passed test.

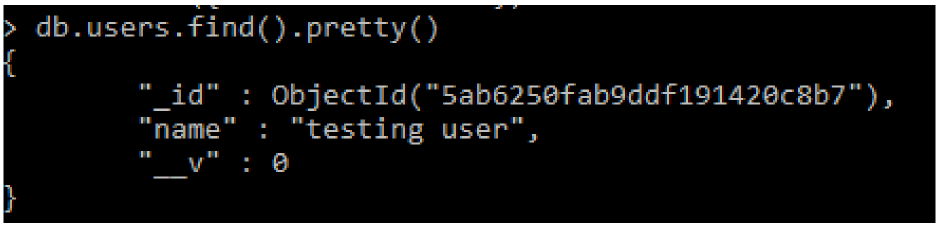


```
✓ master  test pass                          #18 passed
    Commit 581fa0a                           Ran for 1 min 20 sec
    Compare fa4497a..581fa0a                 less than a minute from now
    Branch master

    khoabui9 authored and committed
```

```
$ npm test

> reactexpress@1.0.0 test /home/travis/build/khoabui9/chat
> mocha "server/test"



  save and find Record
Successfully connected to MongoDB!
    ✓ Should return data saved to the database


  1 passing (112ms)
```

Figure 23. Successful test report

The user created from the test case is saved in the database in object format including the id of the user which was auto-generated by MongoDB, and the name of the user. The following figure demonstrates the object user saved in the database.

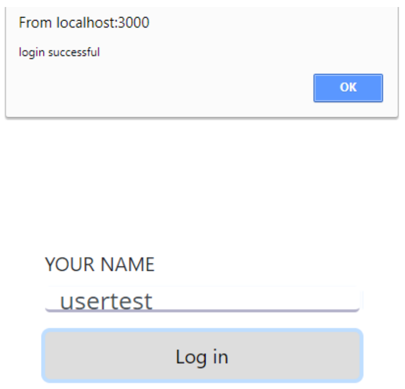


```
> db.users.find().pretty()
{
        "_id" : ObjectId("5ab6250fab9ddf191420c8b7"),
        "name" : "testing user",
        "__v" : 0
}
```

Figure 24. User data saved in the database

To see how the function works in the client-side which is the actual application will be delivered to the client, it is necessary to set up the API which will send the request for adding data or retrieving data and call the API from the client-side. This step is simple since the function was working correctly due to the test report.

In the EasyChat application, when the user logs in, a dialog box will show and inform the user whether the user login successful or login failed. The function “saveUser” works appropriately according to the test report, and the figure 24 shows that the function worked since the user login successfully.

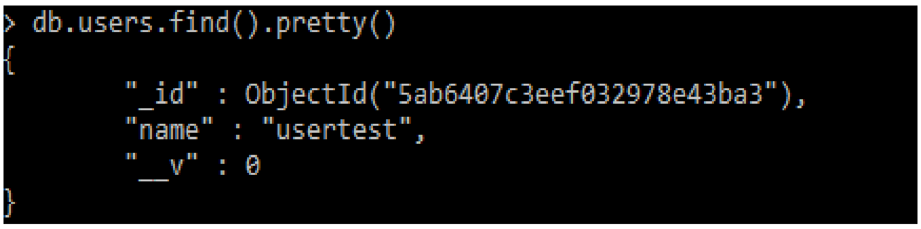


From localhost:3000

login successful

Figure 25. Login successful dialog box

In addition, when signing in to the application, the user “usertest” will be saved in the database, which will prove that the function works as expected. The below figure shows that the function worked correctly since the data is saved in the database.



```
> db.users.find().pretty()
{
        "_id" : ObjectId("5ab6407c3eef032978e43ba3"),
        "name" : "usertest",
        "__v" : 0
}
```

Figure 26. User saved successfully in the database

After logging in successfully, it redirects the user to the main page of the EasyChat application. In the left sidebar, it shows the name of the user just logged in to the application.

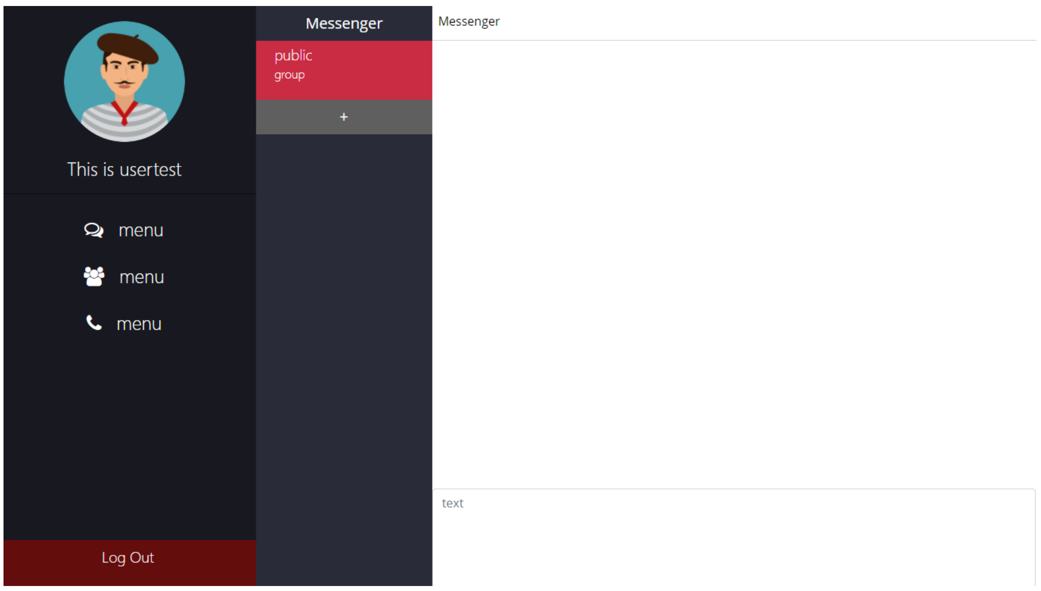
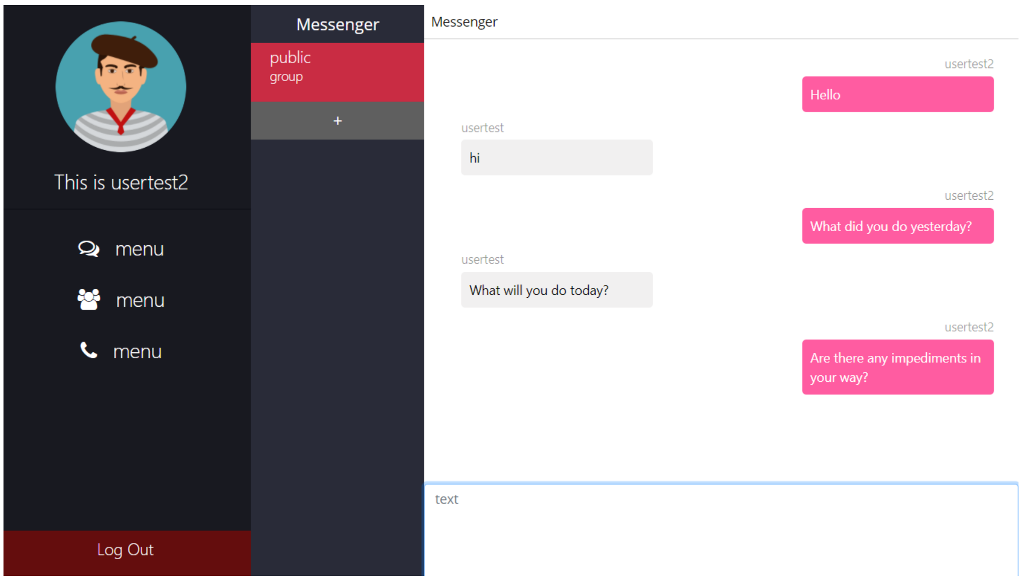


Figure 27. The EasyChat application

The developer applied the same development workflow to other requirements of the EasyChat application using Scrum and test-driven development technique. Since the above function is simple; it is not necessary to refactor. However, in different functions in the application, sometimes it is needed to apply refactoring step in TDD which will remove the non-functional attribute without changing the behavior of the function.

5.7 Project results

At the end of the project, the author was able to finish all the requirements and the specifications of the application on schedule. Furthermore, by applying test-driven development, the author identified some benefits of TDD and how it improved the development process. The following figure demonstrates a conversation in the application.



usertest2
Hello

usertest
hi

usertest2
What did you do yesterday?

usertest
What will you do today?

usertest2
Are there any impediments in your way?

Figure 28. Screenshots of EasyChat application

6 THE STUDY

This section compares the results of the EasyScrum application development process and the EasyChat application development process based on two criteria: time and quality. The author takes the example of checklist feature and user functions to compare two projects.

Quality

A factor that differentiates the EasyScrum and the EasyChat project is the quality. For instance, in the below figure, it is the checklist feature with no checkbox and the delete buttons that did not work.

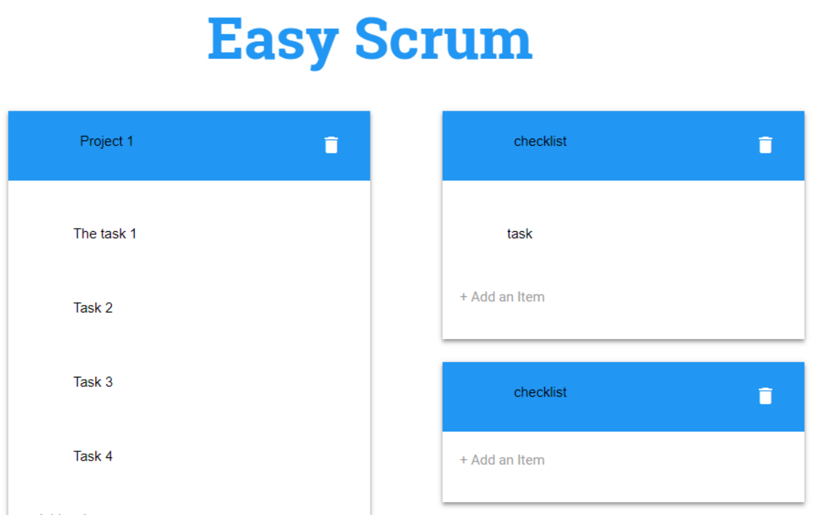
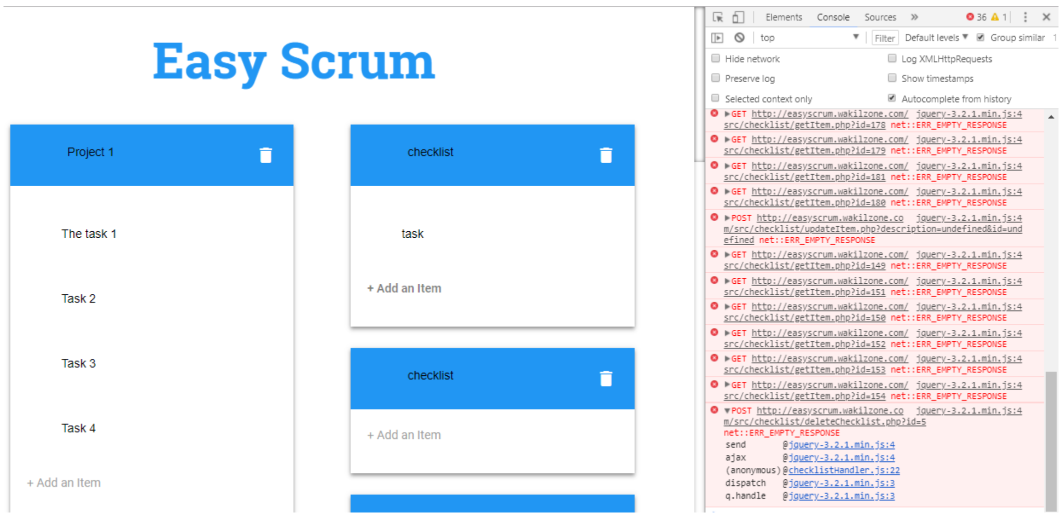


Figure 29. Checklist feature of EasyScrum application

After realizing there are a lot of problems in the application, the developer tried to find exactly why the features did not work by rechecking the code or use the “inspect” feature of Google Chrome to see what the errors are. Those testing methods did help much because it does not show the error and the problem precisely enough. The following figure demonstrates that the browser only shows the ERR_EMPTY_RESPONSE error.



Hide network · Log XMLHttpRequests · Preserve log · Show timestamps · Selected context only · Autocomplete from history [✓]

```
▶GET http://easyscrum.wakilzone.com/  jquery-3.2.1.min.js:4
src/checklist/getItem.php?id=178 net::ERR_EMPTY_RESPONSE
▶GET http://easyscrum.wakilzone.com/  jquery-3.2.1.min.js:4
src/checklist/getItem.php?id=179 net::ERR_EMPTY_RESPONSE
▶GET http://easyscrum.wakilzone.com/  jquery-3.2.1.min.js:4
src/checklist/getItem.php?id=181 net::ERR_EMPTY_RESPONSE
▶GET http://easyscrum.wakilzone.com/  jquery-3.2.1.min.js:4
src/checklist/getItem.php?id=180 net::ERR_EMPTY_RESPONSE
▶POST http://easyscrum.wakilzone.co   jquery-3.2.1.min.js:4
m/src/checklist/updateItem.php?description=undefined&id=und
efined net::ERR_EMPTY_RESPONSE
▶GET http://easyscrum.wakilzone.com/  jquery-3.2.1.min.js:4
src/checklist/getItem.php?id=149 net::ERR_EMPTY_RESPONSE
▶GET http://easyscrum.wakilzone.com/  jquery-3.2.1.min.js:4
src/checklist/getItem.php?id=151 net::ERR_EMPTY_RESPONSE
▶GET http://easyscrum.wakilzone.com/  jquery-3.2.1.min.js:4
src/checklist/getItem.php?id=150 net::ERR_EMPTY_RESPONSE
▶GET http://easyscrum.wakilzone.com/  jquery-3.2.1.min.js:4
src/checklist/getItem.php?id=152 net::ERR_EMPTY_RESPONSE
▶GET http://easyscrum.wakilzone.com/  jquery-3.2.1.min.js:4
src/checklist/getItem.php?id=153 net::ERR_EMPTY_RESPONSE
▶GET http://easyscrum.wakilzone.com/  jquery-3.2.1.min.js:4
src/checklist/getItem.php?id=154 net::ERR_EMPTY_RESPONSE
▼POST http://easyscrum.wakilzone.co   jquery-3.2.1.min.js:4
m/src/checklist/deleteChecklist.php?id=5
net::ERR_EMPTY_RESPONSE
 send        @jquery-3.2.1.min.js:4
 ajax        @jquery-3.2.1.min.js:4
 (anonymous) @checklistHandler.js:22
 dispatch    @jquery-3.2.1.min.js:3
 q.handle    @jquery-3.2.1.min.js:3
```

Figure 30. Checklist feature errors

In EasyScrum project, since the test is done manually, and the test is done in the end after finishing writing the production code, the author could not test all the components at the same time or cannot find any problems within a sprint. Therefore, the application after a sprint usually did not work or did not work as expected.

On the contrary, during the development process of the EasyChat application, the author had to focus and understand what the desired result and how the test should be to write the test and then based on the test, the author will develop the functions. In addition, there are many test cases in the project, and every change required running all the test again to see if there are any problems or did the changes affect other components. Also, the author received the feedbacks within minutes, and the feedback from the testing framework shows the error and the reasons, the expected and the actual result precisely. Therefore, it is easy to find out the problems and fix it. The author also realized that it is safer to refactor since the tests are already passed. The refactoring step in TDD also helps the author to clean and improve the design and the code of the application so that the code will be maintainable, flexible and easily extensible. As mentioned above in chapter 5, the application worked as expected and all tests passed.

Time

After involving in two development process of the above cases, the author realized there are some improvements in the development time of EasyChat project which applied TDD along with Scrum compared to the EasyScrum project which only applied Scrum. As mentioned above, during the EasyScrum project, the author spent days finding the problem, testing again and fixing the bugs due to doing the test at last of the sprint and testing manually. As in the planned project schedule, the expected finished date of checklist features including “checklist backend” and “test checklist” tasks were in 09/10/2017 and 10/10/2017. However, the actual date was in 15/10/2017, and the author was not able to meet the deadline.

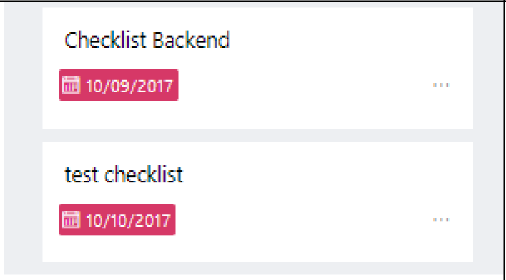
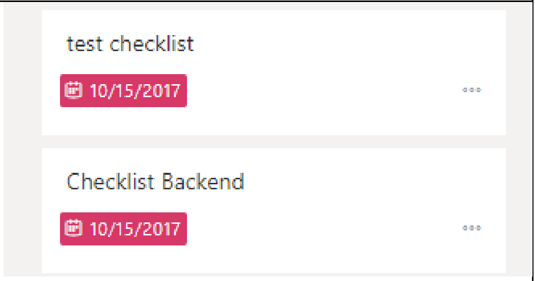
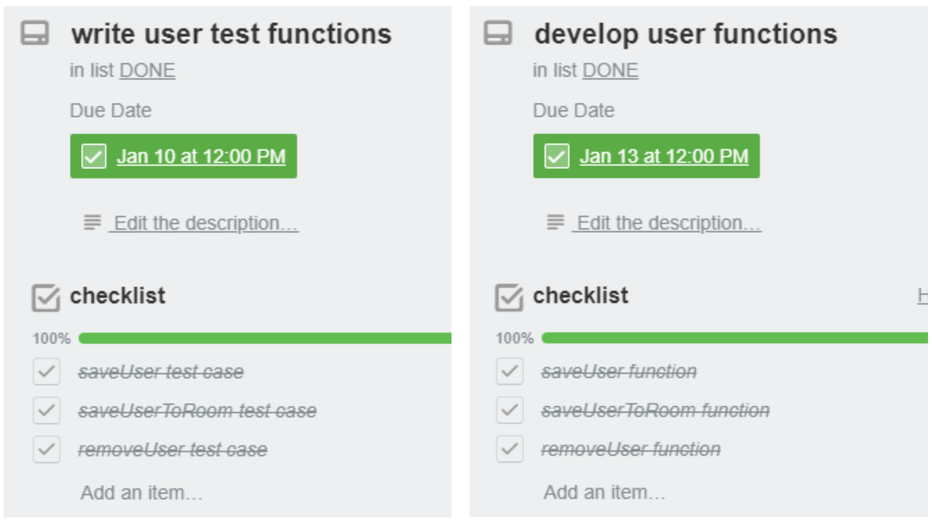
Expected	Actual
 Checklist Backend 10/09/2017 ... test checklist 10/10/2017 ...	 test checklist 10/15/2017 ... Checklist Backend 10/15/2017 ...

Figure 31. Expected and actual finished date of checklist feature

In contrast, by applying test-driven development in Scrum, the author witnessed a significant improvement in time in the EasyChat project. Since the testing in TDD provides immediate feedback, the author can quickly identify the problems and the bugs to respond and make changes as soon as the problems occur. Moreover, when doing the test first and then developing the function, the author can save time by writing just a small amount of code to make the test pass. It is clear that the code is finished when the test pass and the author can move to the next development task.


write user test functions
in list DONE
Due Date
Jan 10 at 12:00 PM
Edit the description…
checklist
100%
~~saveUser test case~~
~~saveUserToRoom test case~~
~~removeUser test case~~
Add an item…

develop user functions
in list DONE
Due Date
Jan 13 at 12:00 PM
Edit the description…
checklist
100%
~~saveUser function~~
~~saveUserToRoom function~~
~~removeUser function~~
Add an item…

Figure 32. User functions and User test functions tasks

The sprint started from 8/1/2018 and ended in 14/1/2018, and two tasks and all the items in the checklist were marked as done in 10/1/2018 and 13/1/2018. The author was able to finish the task before the deadline and have time for sprint retrospective and prepare for the next sprint.

7 CONCLUSIONS AND FURTHER RESEARCH

The concluding chapter of this thesis contains four sections. First, the author answers the research question. The next two sections discuss the limitations, the reliability and the validity of this thesis. The last section suggests ideas for future research.

7.1 Answering research question

The goal of this thesis was to identify the benefits of applying test-driven development in Scrum. The theoretical part provided a knowledge base and understanding of basic methodologies and technologies including Scrum, test-driven development as well as the testing framework. Moreover, the result from the comparison of the EasyScrum application and EasyChat gives information to answer the research question:

What are the benefits of the test-driven development technique and how could it be applied to improve Scrum?

Applying the test-driven development technique along with Scrum brings many benefits to teams by improving the development process of the project. The author aimed to answer the research question by comparing the development process of two projects, and the improvements which can be seen from the study are time and quality.

The test-driven development technique has the greatest impact on the quality of the final product. There are different test cases for every function project, and the test cases provide the feedbacks within minutes. Therefore, the author can know what should be changed and fixed. Moreover, when the test pass, the author came back to refactor the code which removes the duplication so that the codes have a clean design and easier to understand. In short, by using TDD, the code is maintainable and flexible, and the quality of the code is improved.

Test-driven development helps the author to save time and use the time more efficient. During the project, the automated test provides immediate feedback so that the author can react quickly to the problems and make changes as soon as the problems occur. Although, writing a small amount of code to make test pass also helps the developer to save time.

7.2 Limitations

The result of the study is based on the comparison of the development process of the EasyScrum project and the EasyChat project. The author developed the EasyChat application and studied the concept of TDD and the testing framework in a short amount of time. Therefore, the author may have missed some criteria to explain more about the differences between two processes and may not have discovered all benefits of applying TDD in Scrum.

Since the data was collected by the author observing of the development process of two projects, the author may have not collected enough data to identify differences and the data may not be fully suitable or applicable for the study.

7.3 Reliability and Validity

The research data analyzed in the thesis was collected by observing the development processes during two projects. Therefore, the reliability and the validity of the study are measured based on the result of the study. For the repeatability of findings, the result may not be wholly the same, but the benefits of test-driven development can still be seen in addition to time and quality. Additionally, the size of the project, the technology used, the duration of the project and the complexity of the code or the ability of the author can affect the result of the study

7.4 Suggestions for further study

After working on this research study about the benefit of test-driven development, further study could be based the comparison between larger scale projects or more complex projects. Those projects can have more complex test cases which may fully show the benefits and the limitations of TDD workflow.

Another future work could be studying different features of Mocha – the testing framework used in this study or another testing frameworks which can support TDD better.

8 SUMMARY

During the past years, agile development methodologies have evolved fast in the technology world. Teams and organization have been collaborating and working based on those agile methodologies. The most used agile practice is Scrum methodology which brings a lot of benefits, and test-driven development is one of the famous software development techniques which is often applied in Scrum. The goal of this study is to identify the benefit of test-driven development bring to Scrum by comparing the new artifact applied TDD technique with Scrum created by the author and the previous project which only applied Scrum.

This study comprises of three sections: Theoretical framework, practices and the study.

Chapter 3 covers the theoretical framework. In Chapter 3, the author includes the concept of agile development methodologies, Scrum, and test-driven development. Along with the concepts, the information of related technologies and tools are also provided. Chapter 4 and Chapter 5 describes the development process of two projects for comparing purpose. Chapter 6 carries the result from the comparison.

In conclusion, all three sections of this thesis allow the author to answer the research question. Additionally, the limitations of the study and further research suggestions are included.

LIST OF REFERENCES

Published Sources

Ashmore, S.& Runyan, K. 2015. Introduction to Agile Methods. United States: Pearson Education, Inc.

Bender, J. & McWherter J. 2011. Professional Test-Driven Development with C#: Developing Real World Applications with TDD. Canada: Wiley Publishing, Inc.

Harwell, M. 2011. Chapter 10 | Research Design in Qualitative/Quantitative/Mixed Methods in The SAGE Handbook for Research in Education: Pursuing Ideas as the Keystone of Exemplary Inquiry. 2nd ed. California: Sage publications.

Hevner, A. & Chatterjee, S. 2010. Design Research in Information Systems Theory and Practice. New York: Springer.

Krol, J. 2014. Web Development with MongoDB and Node.js. Birmingham: Packt Publishing Ltd.

Pham, A & Pham P.V. 2012. Scrum in Action: Agile Software Project Management and Development. Boston: Course Technology.

Punch, K. F. 2013. Introduction to Social Research: Quantitative and Qualitative Approaches. 3rd ed. London: Sage publications.

Rai, R. 2013. Socket.io Real-time Web Application Development. Birmingham: Packt Publishing Ltd.

Resnick, S. De la Maza, M. & Bjork, A. 2011. Professional Scrum with Team Foundation Server 2010. Canada: Wiley Publishing, Inc.

Saunders, M., Lewis, P. & Thornhill, A. 2012. Research Methods for Business Students. 6th ed. Harlow, England: Pearson Education Limited.

Somasundaram, R. 2013. Git: Version Control for Everyone. Birmingham: Packt Publishing Ltd.

Electronic Sources

Agarwal, S. 2018. 6 Compelling Benefits of (TDD) Test Driven Development. knowledgehut [accessed 10 February 2018]. Available at: https://www.knowledgehut.com/blog/agile-management/6-compelling-benefits-of-tdd-test-driven-development

Atlassian 2018. What is Git?. Atlassian [accessed 5 March 2018]. Available at https://www.atlassian.com/git/tutorials/what-is-git#security

Austin, Z. 2014. Qualitative Research: Getting Started. NCBI [accessed 10 February 2018]. Available at: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4275140/

AWS 2018. What is NoSQL. Amazon [accessed 6 March 2018]. Available at https://aws.amazon.com/nosql/

Barbara B. Kawulich. 2005. Participant Observation as a Data Collection Method [accessed 10 February 2018]. Available at http://home.sogang.ac.kr/sites/kylee/Courses/Lists/b6/Attachments/21/Participant%20Observation%20as%20a%20Data%20Collection%20Method%20(2005).pdf

Elmansy, R. 2018. Using Inductive Reasoning in User Experience Research. Designorate [accessed 06 February 2018]. Available at: http://www.designorate.com/inductive-reasoning-in-user-experience-research/

GitHub 2018. About. Github [accessed 5 March 2018]. Available at https://github.com/about

guru99 2018. Test Driven Development (TDD): Learn with Example. Guru99 [accessed 15 February 2018]. Available at: https://www.guru99.com/test-driven-development.html

Mehra, A. 2016. A Testing Strategy in Scrum. Scrum Alliance [accessed 03 February 2018]. Available at: https://www.scrumalliance.org/community/articles/2016/july/a-testing-strategy-in-scrum

Mochajs 2018. Mochajs introduction. Mochajs.org [accessed 7 March 2018]. Available at https://mochajs.org

MongoDB 2018. Introduction to MongoDB. Mongodb [accessed 6 March 2018]. Available at https://docs.mongodb.com/manual/introduction/#introduction-to-mongodb

Morgan, A. 2017. Introducing the MEAN and MERN stacks. Mongodb [accessed 6 March 2018]. Available at https://www.mongodb.com/blog/post/the-modern-application-stack-part-1-introducing-the-mean-stack

Mozilla 2018. Express/Node introduction. Mozilla [accessed 6 March 2018]. Available at https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction

NodeJS 2018. ECMAScript 2015 (ES6) and beyond. Node.js foundation [accessed 6 March 2018]. Available at https://nodejs.org/en/docs/es6/

Palermo, J. 2006. Guidelines for Test-Driven Development. Microsoft [accessed 16 February 2018]. Available at https://msdn.microsoft.com/en-us/library/aa730844(v=vs.80).aspx

Reactjs 2018. Tutorial: Intro to React. Facebook [accessed 6 March 2018]. Available at https://reactjs.org/tutorial/tutorial.html

Scrum 2018. What is a Sprint Backlog?. Scrum.org [accessed 12 February 2018]. Available at: https://www.scrum.org/resources/what-is-a-sprint-backlog

Sherman, K. 2017. Testing in Agile: The Sprint Is Too Short. Testing Excellence [accessed 03 February 2018]. Available at: https://www.testingexcellence.com/testing-agile-sprint-short/

Soares, M. 2016. Are you more productive and effective with SCRUM?. Linkedin [accessed 10 March 2018]. Available at https://www.linkedin.com/pulse/you-more-productive-effective-scrum-ricardo-macedo-soares/

VersionOne 2018. What Is Scrum Methodology?. VersionOne [accessed 15 February 2018]. Available at: https://www.versionone.com/agile-101/what-is-scrum/

VersionOne. 2018. VersionOne 11th Annual State of Agile Report. VersionOne [accessed 2 February 2018]. Available at: http://www.agile247.pl/wp-content/uploads/2017/04/versionone-11th-annual-state-of-agile-report.pdf

Winter, D. 2016. 9 Benefits of Test Driven Development. Madetech [accessed 15 February 2018]. Available at https://www.madetech.com/blog/9-benefits-of-test-driven-development

APPENDICES

Appendix 1. .travis.yml file for TravisCI configuration

```
language: node_js

node_js:

  - "stable"

services:

  - mongodb

before_script:

  - sleep 15

  - mongo mydb_test --eval
'db.createUser({user:"travis",pwd:"test",roles:["readWrite"]});'
```

Appendix 2. package.json scripts configuration

```
{
  "name": "chat",
  "version": "1.0.0",
  "description": "chat",
  "main": "server/server.js",
  "scripts": {
    "client-install": "cd client && npm install",
    "start": "node server/server.js",
    "server": "nodemon server/server.js",
    "client": "npm start --prefix client",
    "dev": "concurrently \"npm run server\" \"npm run client\"",
    "test": "mocha \"server/test\""
  },
```