ChatGPT as a Fullstack Web Developer - Early Results

Pekka Abrahamsson[1], Tatu Anttila[1], Jyri Hakala[1], Juulia Ketola[1], Anna Knappe[1], Daniel Lahtinen[1], Väinö Liukko[1], Timo Poranen[1](✉), Topi-Matti Ritala[1], and Manu Setälä[2]

[1] Tampere University, Tampere, Finland
[2] Solita Ltd., Helsinki, Finland

Abstract. The arrival of ChatGPT has caused a lot of turbulence also in the field of software engineering in the past few months. Little is empirically known about the capabilities of ChatGPT to actually implement a complete system rather than a few code snippets. This paper reports the first-hand experiences from a graduate level student project where a real-life software platform for financial sector was implemented from the scratch by using ChatGPT for all possible software engineering tasks. The main conclusions drawn are as follows: 1) these findings demonstrate the potential for ChatGPT to be integrated into the software engineering workflow, 2) it can be used for creating a base for new components and for dividing coding tasks into smaller pieces, and 3) noticeable enhancements in ChatGPT-4, compared to ChatGPT-3.5, indicate superior working memory and the ability to continue incomplete responses, thereby leading to more coherent and less repetitive dialogues.

Keywords: AI assisted · software development · software engineering · AI programming · ChatGPT · large language models · artificial intelligence

1 Introduction

The introduction of ChatGPT into the landscape of technology has generated a notable amount of disruption, especially within the field of software engineering. However, despite this growing interest, empirical knowledge about the actual capabilities of ChatGPT remains limited. This lack of comprehensive understanding is particularly evident when considering the potential of ChatGPT to design and implement holistic systems as opposed to merely generating discrete fragments of code. There exists a significant difference between crafting isolated code snippets and deploying a fully realized software solution, a distinction that is yet to be thoroughly explored in the context of ChatGPT.

This article describes a student software project that was created to explore the use of artificial intelligence (AI) in software development. The main goal of the project was to investigate the effectiveness of ChatGPT in practice.

Overall, this project contributes to the field of AI-assisted software development by providing valuable experience of using ChatGPT as tool in software development. The remainder of this article provides related research in Sect. 2, a detailed description of the project and the research design in Sect. 3, results in Sect. 4, and conclusions.

2 AI Assisted Software Development

Artificial Intelligence (AI) is a branch of computer science that focuses on creating intelligent machines that can perform tasks that typically require human intelligence, such as understanding natural language, recognizing patterns, making decisions, and solving problems.

One type of AI tool that has gained significant attention in recent years is the large language model (LLM) like OpenAI's GPT-4 [7]. LLM is AI model that is trained on massive amounts of data to generate human-like text output. GPT-4 uses transformer-style model to predict the content and structure of text based on an input, usually text as well. This can be used in a wide range of natural language processing tasks, such as language translation, text summarization, and answering questions.

ChatGPT has provided a chatbot interface for interacting with OpenAI's GPT-models [2]. This has made the capabilities of LLMs more widely known, which in turn has sparked the research around use cases for this technology. Current research include studies related to prompt patters [13], human-bot collaborative architecting [1] and using ChatGPT for programming numerical methods [6]. Treude [11] has developed a prototype to compare different GPT model solutions, and Dong and others [4] developed a self-collaboration code generation framework. Surameery and Shakor [10] have applied ChatGPT to solve programming bugs.

3 Research Design

In this section we introduce the project background and project implementation details including the implemented features.

3.1 Project Background

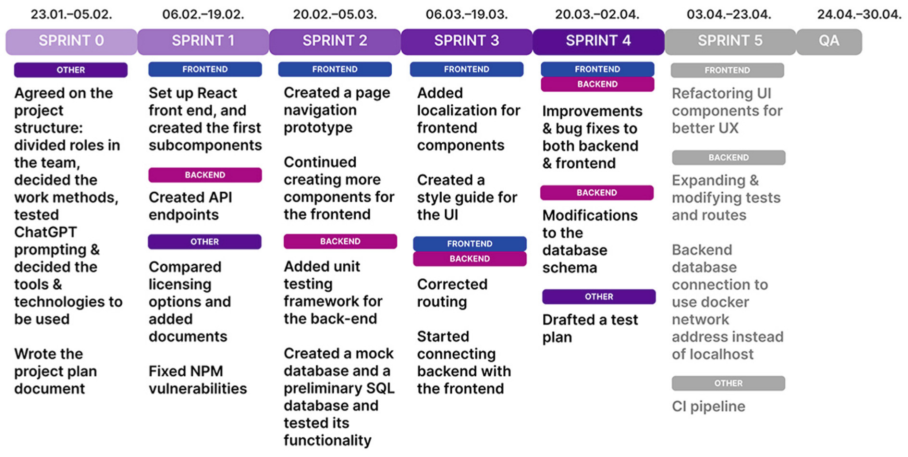
Solita Ltd. [8] is a large software consultancy company in the Nordic countries. Solita collaborates with universities by inventing exercise topics and supervising student exercises. They challenged the student team to undertake an AI-assisted, large-scale project.

Project topic was chosen from well-defined public procurement requests on the Hilma portal [5], a website for procurement in the Finnish public sector. It was agreed in the project that the specifications would not be directly used as input material for AI, but the prompts given to AI were written mostly by the team themselves. However, the number of fields in the user interface was kept the same as in the original request, etc.

The selected project, Valvontatyöpöytä (VTP) is a platform for financial supervision, designed to support the operations of an organization. The intended user group for the VTP is financial professionals, including supervisors, managers, and analysts.

3.2 Project Implementation

The VTP project [12] was proposed in the end of December 2022. The project was then accepted by a seven member team. The project started in the end of January 2023. The team consists of three master's level students and four Bachelor's level students of Computer science or Information technology. None of the team members had earlier experience on AI assisted software development. General overview of the course's practices and schedule are described by Sten and others [9]. First was sprint 0 (one week) to plan the project and set up the development environment, then five two-week implementation sprints, and then one week quality assurance sprint. The total duration of the project is 15 weeks. Sprint 5 and the QA sprint are not covered in this report as the project is ongoing. Project phases and implemented features per sprint are show in Fig. 1. The team utilized a Kanban board to manage tasks and issues.



23.01.–05.02. SPRINT 0
OTHER
Agreed on the project structure: divided roles in the team, decided the work methods, tested ChatGPT prompting & decided the tools & technologies to be used
Wrote the project plan document

06.02.–19.02. SPRINT 1
FRONTEND
Set up React front end, and created the first subcomponents
BACKEND
Created API endpoints
OTHER
Compared licensing options and added documents
Fixed NPM vulnerabilities

20.02.–05.03. SPRINT 2
FRONTEND
Created a page navigation prototype
Continued creating more components for the frontend
BACKEND
Added unit testing framework for the back-end
Created a mock database and a preliminary SQL database and tested its functionality

06.03.–19.03. SPRINT 3
FRONTEND
Added localization for frontend components
Created a style guide for the UI
FRONTEND
BACKEND
Corrected routing
Started connecting backend with the frontend

20.03.–02.04. SPRINT 4
FRONTEND
BACKEND
Improvements & bug fixes to both backend & frontend
BACKEND
Modifications to the database schema
OTHER
Drafted a test plan

03.04.–23.04. SPRINT 5
FRONTEND
Refactoring UI components for better UX
BACKEND
Expanding & modifying tests and routes
Backend database connection to use docker network address instead of localhost
OTHER
CI pipeline

24.04.–30.04. QA

Fig. 1. Project phases and implemented features per sprint.

3.3 Development Environment

The development environment was built piecemeal by consulting ChatGPT on suitable technologies for the projects needs. The team fed ChatGPT prompts explaining what they were currently trying to accomplish and ChatGPT replied with multiple recommendations. The team then cherry-picked from the recommendations based on their own preferences and previous experience. The effect of picking technologies with such a process is twofold. Firstly, ChatGPT is more likely to recommend technologies it has knowledge of. Secondly, it makes the teams' efforts in reviewing the generated code easier.

Table 1. Technical implementation environment and technology selection criteria.

Item	Description	Selection criteria
Language	JavaScript	Recommended by ChatGPT
Language	HTML and CSS	Recommended by ChatGPT
Language	Shell	Developer decided
Containerization	Docker	Customer requirement
Database	MySQL	Recommended by ChatGPT
AI assistant	ChatGPT 3.5 and 4.0	Customer requirement
Front-end framework	React	Recommended by ChatGPT
Back-end framework	Node.JS, Express	Recommended by ChatGPT
Test framework	Mocha, Chai	Recommended by ChatGPT
Version control system	GitHub	Team decided
Licence	MIT	Recommended by ChatGPT

Table 1 lists technologies used in the project with the corresponding selection criteria. As seen in Table 1 there are exceptions on which ChatGPT was not consulted at all. These include using ChatGPT as the sole AI assistant, and containerization of the produced application, both of which were requirements laid out by the customer. The team decided to use GitHub as their version control system to enable easy collaboration. The only exception in languages used in the project comes from a Bash script used by the backend container to wait for the database to fully initialize before trying to establish a connection.

3.4 Development Process

The team's process of working with ChatGPT is illustrated in Fig. 2. When a task was related to existing code, the assigned team member would provide the relevant code to ChatGPT and request it to generate a solution. If the task was not related to existing code, the team member would first ask ChatGPT for recommendations before requesting it to produce the code. Once ChatGPT generated the code, the team member would review it for correctness. If the code was deemed satisfactory, the team member would add it to the code base, save the chat session as a markdown file, and create a pull request with the chat as an attachment. If the code was not acceptable, the team member would provide the problematic code back to ChatGPT and repeat the process, iterating until a satisfactory solution was achieved or asking for a new recommendation for another approach.

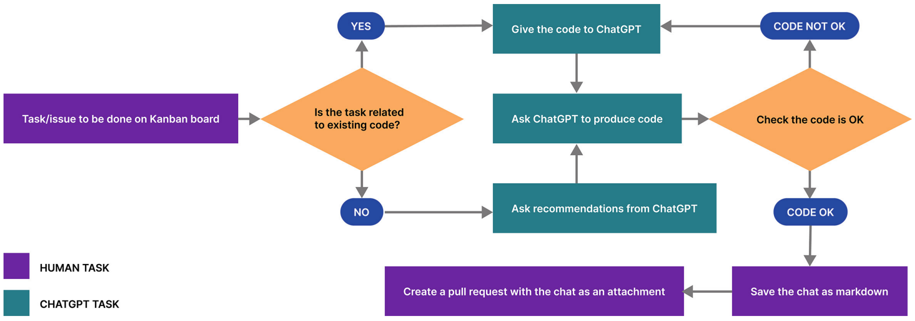

YES
Give the code to ChatGPT
CODE NOT OK
Task/issue to be done on Kanban board
Is the task related to existing code?
Ask ChatGPT to produce code
Check the code is OK
Ask recommendations from ChatGPT
NO
CODE OK
HUMAN TASK
CHATGPT TASK
Create a pull request with the chat as an attachment
Save the chat as markdown

Fig. 2. Process of working with ChatGPT.

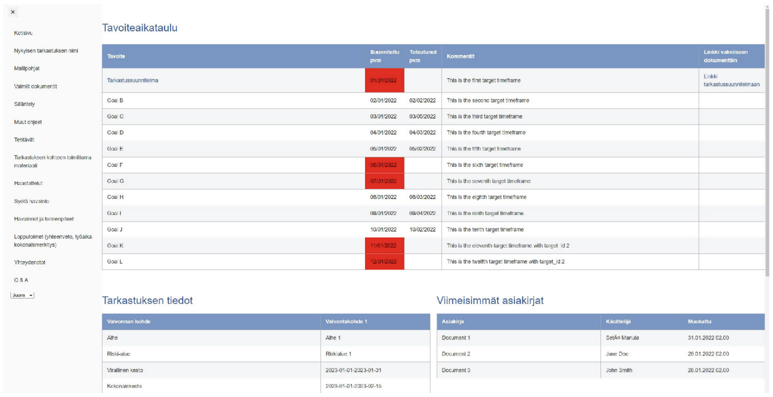
The project team logged their weekly workings hours according to different categories (Documentation, Requirements, Design, Implementation, Testing, Meetings, Studying, Other, Lectures). After sprint 4 the project had a total 607 h logged. Throughout the project, the project team met with the customer twice a week to ensure quality assurance and planning were on track. This is a reason why meetings (198 h, 33%) have such a considerable part in the logged hours. So far the implementation has taken 131 h (22%) and studying 116 h (19%). The logged time also included university course related subjects, such as lectures and other studying, which did not relate directly to the implementation of the project.

3.5 Implemented Features

Based on ChatGPT's suggestion, we began by setting up the foundation for our React JavaScript project. Implemented features per sprints are shown in Fig. 1. Following the wireframes in the original procurement requests, we commenced implementing the Inspection Information page shown in Fig. 3. We adopted a component-by-component approach to building the application, and once the first frontend components were completed, we proceeded to develop the backend and database infrastructure.

To better comprehend the application's functionality, we utilized the wireframe images to create a prototype. In our development process, we proceeded to implement a new view for the application, the Inspection Plan page. Additionally, we created localization possibility into the application, allowing for text to be displayed in both Finnish and English. With the assistance of ChatGPT, we created a style guide for the UI and began implementing it into the application.

We proceeded to build the correct routing and to integrate the frontend and backend together. Recognizing the need for adjustments to the database


×
Kotisivu
Nykyisen tarkastuksen nimi
Mallipohjat
Valmiit dokumentit
Sääntely
Muut ohjeet
Tehtävät
Tarkastuksen kohteen toimittama materiaali
Haastattelut
Syötä havainto
Havainnot ja toimenpiteet
Lopputoimet (yhteenveto, työaika, kokonaismerkitys)
Yhteydenotot
Q & A
Suomi

Tavoiteaikataulu

Tavoite	Suunniteltu pvm	Toteutunut pvm	Kommentit	Linkki valmiiseen dokumenttiin
Tarkastussuunnitelma	01/01/2022		This is the first target timeframe	Linkki tarkastussuunnitelmaan
Goal B	02/01/2022	02/02/2022	This is the second target timeframe	
Goal C	03/01/2022	03/05/2022	This is the third target timeframe	
Goal D	04/01/2022	04/03/2022	This is the fourth target timeframe	
Goal E	05/01/2022	05/02/2022	This is the fifth target timeframe	
Goal F	06/01/2022		This is the sixth target timeframe	
Goal G	07/01/2022		This is the seventh target timeframe	
Goal H	08/01/2022	08/03/2022	This is the eighth target timeframe	
Goal I	09/01/2022	09/04/2022	This is the ninth target timeframe	
Goal J	10/01/2022	10/02/2022	This is the tenth target timeframe	
Goal K	11/01/2022		This is the eleventh target timeframe with target_id 2	
Goal L	12/01/2022		This is the twelfth target timeframe with target_id 2	

Tarkastuksen tiedot

Valvonnan kohde	Valvontakohde 1
Alhe	Alhe 1
Riski-alue	Riskialue 1
Virallinen kesto	2023-01-01-2023-01-31
Kokonaiskesto	2023-01-01-2023-02-15

Viimeisimmät asiakirjat

Asiakirja	Käsittelijä	Muokattu
Document 1	SetÃ¤ Manula	31.01.2022 02.00
Document 2	Jane Doe	29.01.2022 02.00
Document 3	John Smith	28.01.2022 02.00

Fig. 3. Inspection information in VTP.

schema, we initiated revisions while concurrently addressing application styling and adding several popup forms.

4 Results

In this section we observe ChatGPT discussions, code and lessons learned.

4.1 Discussions with ChatGPT

We stored ChatGPT conversations that were relevant to the project in Markdown format. The resulting Markdown files were included into the description part of pull requests made to the project GitHub repository.

The conversation length varies a lot depending on how big changes were requested and if any problems occurred. From sprint 0 to sprint 4, 39 pull requests were merged to the codebase. In these conversations the number of prompts made ranged from 1 to 129. On average a pull request had 19 prompts. A common way to start conversation with ChatGPT was to describe the problem or wanted feature or to paste existing code and ask for needed changes. If the AI model couldn't give a solution directly based on the first prompt, the developer would give more information or clarifications in an iterative manner where the solution was found by fine-tuning the earlier answers.

4.2 Code

The application implements a basic three-tier architecture, where the front-end communicates with the back-end through a REST API. Each tier runs in their own Docker container and the whole system is managed with Docker Compose.

Table 2 provides a breakdown by type of file and line for the project's code base. File and blank, comment and code line counts were calculated from the project's repository using cloc [3]. The vast majority of lines shown are a result of bootstrapping a React project to be used as the front-end client. This includes most of the JSON lines that node package manager uses for managing dependencies.

However, the team managed to use ChatGPT to generate all of the code directly related to running the project. This includes containerization (both Dockerfiles, a YAML file), CI pipeline (other YAML file), database initialization clauses (SQL), and the entire RESTful API that makes up the project's back-end, including unit tests for its routes. Furthermore, all React components, their layout and styling (CSS) were created by ChatGPT based on the teams instructions. The HTML files contain a style guide. In total, 4000 lines of code were generated by ChatGPT.

Table 2. Output of the cloc [3] package based on the project's repository as of 2.4.2023.

Language	files	blank	comment	code
JSON	6	0	0	18932
JavaScript	36	259	68	2798
CSS	15	149	22	796
HTML	3	42	30	357
Bourne Shell	1	12	6	164
Markdown	4	51	0	82
SQL	1	12	0	73
YAML	2	9	0	58
Dockerfile	2	14	3	20
SVG	1	0	0	1
SUM:	**71**	**548**	**129**	**23281**

In general, ChatGPT produces code that appears to be relatively high quality. The code mostly works, it is laid out in a logical manner, and has well named variables that make it easy to understand. Its two biggest pitfalls are consistency and attention to detail. The former shows as stylistic differences in blocks of code produced in separate replies by ChatGPT, which were sometimes incompatible to the point of not functioning. The latter problem was mainly encountered when attempting to fit pieces of the project together as it grew more complex which meant conversations with ChatGPT needed more context to be provided. The team noticed marked improvement in both areas when using GPT-4 over GPT-3.5.

4.3 Lessons Learned

ChatGPT has both strengths and limitations in assisting with software development tasks. While it can be helpful in creating a base for new components, fine-tuning and getting the final result may take longer. It may struggle with updating code consistently, remembering to make changes, and handling errors. Additionally, it can misunderstand questions, have issues with non-determinism, and suffer from token limitations in answers. However, breaking requests into smaller pieces, providing clear descriptions, and using various techniques to avoid length restrictions can improve its effectiveness.

One team member described experience of working with ChatGPT like a rubber ducking, but the duck actually responds. In similar vein to rubber ducking, the output from the language model depends on how well the developer is able to express the problem at hand. A Clear well defined prompt would return better code than a prompt by someone with only a vague idea of what they were trying to achieve. Good development practises still apply, even if the code is written by AI.

5 Conclusions

This paper presents first-hand experiences of using ChatGPT to develop a full-stack software application. Overall, this study contributes to the growing body of literature on the application of language models in software engineering. This study also provides insights for researchers and practitioners interested in exploring the use of ChatGPT for developing real-world software systems.

Based on early results from this exploratory study, the main conclusions drawn were as follows: 1) these findings demonstrate the potential for ChatGPT to be integrated into the software engineering workflow, 2) it can be used for creating a base for new components and for dividing coding tasks into smaller pieces, and 3) noticeable enhancements in ChatGPT-4, compared to ChatGPT-3.5, indicate superior working memory and the ability to continue incomplete responses, thereby leading to more coherent and less repetitive dialogues.

Next steps in the research will include reporting experiences from the remaining sprints, test the application systematically, analyse code quality, and compare ChatGPT generated code with the code written by the developers.

References

1. Ahmad, A., Waseem, M., Liang, P., Fehmideh, M., Aktar, M.S., Mikkonen, T.: Towards human-bot collaborative software architecting with ChatGPT. arXiv preprint arXiv:2302.14600 (2023)
2. ChatGPT. https://chat.openai.com/ (2023). Accessed 6 Apr 2023
3. cloc - Count lines of Code. https://github.com/AlDanial/cloc (2023). Accessed 13 Apr 4 2023
4. Dong, Y., Jiang, X., Jin, Z., Li, G.: Self-collaboration code generation via Chatgpt. arXiv preprint arXiv:2304.07590 (2023)
5. Hilma - Public procurement. https://www.hankintailmoitukset.fi/en/ (2023). Accessed: 31 Mar 2023
6. Kashefi, A., Mukerji, T.: ChatGPT for programming numerical methods. arXiv preprint arXiv:2303.12093 (2023)

7. OpenAI: GPT-4 Technical Report. arXiv preprint arXiv:2303.08774 (2023)
8. Solita company. https://www.solita.fi/en/company/ (2023). Accessed 31 Mar 2023
9. Sten, H., Ahtee, T., Poranen, T.: Evaluation of students' capstone software development projects. In: SEFI Annual Conference, pp. 531–540 (2018)
10. Surameery, N.M.S., Shakor, M.Y.: Use Chat GPT to solve programming bugs. Int. J. Inf. Technol. Comput. Eng. (IJITC) **3**(01), 17–22 (2023). ISSN: 2455-5290
11. Treude, C.: Navigating complexity in software engineering: a prototype for comparing GPT-n solutions. arXiv preprint arXiv:2301.12169 (2023)
12. VTP - Source code repository for the Valvontatyöpöytä. https://github.com/AI-Makes-IT/VTP (2023). Accessed 31 Mar 2023
13. White, J., Hays, S., Fu, Q., Spencer-Smith, J., Schmidt, D.C.: ChatGPT prompt patterns for improving code quality, refactoring, requirements elicitation, and software design. arXiv preprint arXiv:2303.07839 (2023)