# Automation of the Game Master in Escape Rooms


T. van Asten
B.R.A. Bot
B.I.Y.L. Ho
J.R.R. Lopez Küchlin

# Escape-Master

*Automation of the Game Master in Escape Rooms*

Final Report

Bachelor End Project
Computer Science & Engineering

by

**T. van Asten**
**B.R.A. Bot**
**B.I.Y.L. Ho**
**J.R.R. Lopez Kuchlin**

| **Name** | **Email** |
|---|---|
| Timo van Asten | t.vanasten@student.tudelft.nl |
| Bernard Bot | b.r.a.bot@student.tudelft.nl |
| Björn Ho | b.i.y.l.ho@student.tudelft.nl |
| Jael lopez Kuchlin | j.r.r.lopezkuchlin@student.tudelft.nl |
| | |
| Project duration: | April 23, 2018 – July 4, 2018 |
| | |
| TU Coach: | W.P. Brinkman |
| Client Advisor: | J.W. Manenschijn |
| Bachelor Project Coordinators: | O. Visser, H. Wang, M. de Weerdt |

TU Delft

# Contents

**Foreword and Acknowledgements** 1

**Summary** 2

**1 Introduction and research** 3
- 1.1 Introduction . . . 3
- 1.2 Problem definition and analysis. . . . 3
  - 1.2.1 Definition . . . 3
  - 1.2.2 Analysis . . . 4
  - 1.2.3 Gameplay . . . 6
- 1.3 Product goals . . . 6
  - 1.3.1 Interaction Design . . . 6
  - 1.3.2 Requirements . . . 8
- 1.4 Groundwork . . . 9
  - 1.4.1 Automation of hints . . . 10
  - 1.4.2 Modular Architecture . . . 11
  - 1.4.3 Interaction with Creator . . . 12
  - 1.4.4 Communication with Players . . . 13

**2 Design** 16
- 2.1 Initial designs . . . 16
- 2.2 Challenges . . . 17
  - 2.2.1 Watson . . . 17
  - 2.2.2 Messaging . . . 17
  - 2.2.3 Game states . . . 18
  - 2.2.4 Game events . . . 18
- 2.3 Final Design. . . . 18

**3 Implementation** 20
- 3.1 Creator system . . . 20
  - 3.1.1 Creating an escape room . . . 20
  - 3.1.2 Starting an escape room . . . 21
  - 3.1.3 Deployment . . . 22
- 3.2 Communication with players . . . 22
- 3.3 Automation of hints . . . 24
- 3.4 Persistence and modularization. . . . 25

**4 Testing** 26
- 4.1 System testing. . . . 26
  - 4.1.1 IBM Watson API . . . 26
  - 4.1.2 Creator system . . . 26
  - 4.1.3 Unit testing . . . 26
  - 4.1.4 Results . . . 26
- 4.2 Quality assurance . . . 27
  - 4.2.1 Code quality . . . 27
  - 4.2.2 Expert review . . . 27
  - 4.2.3 Field studies . . . 27
  - 4.2.4 Results . . . 30
- 4.3 Discussion . . . 33

**5 Conclusion and discussion** **34**
5.1 Evaluation . . . 34
5.2 Limitations . . . 34
5.3 Ethical implications . . . 35
5.4 Contributions and recommendations . . . 35

**Bibliography** **37**

**Appendices** **39**

**A Project description** **40**

**B Info sheet** **41**

**C Roadmap** **43**

**D SIG Recommendations** **45**
D.1 First feedback . . . 45
D.2 Incorporation of feedback . . . 45
D.3 Second feedback . . . 46

**E UML** **47**

**F Questionnaire** **49**

**G Questionnaire results** **52**
G.1 Field study 1 . . . 52
G.2 Field study 2 . . . 57

**H Project Plan** **62**

# Foreword and Acknowledgements

This report concludes the TI3806 Bachelorproject (2017/18 Q4) course and contains all relevant, related information. It describes the progress during the ten week period of the project spent working for the client, Popup Escape. A product was developed that assisted the hosts of escape rooms with the automation of hints. Furthermore, this document will elucidate the completed work and part of the accompanying process. Finally, it shall provide recommendations for future work on this type of system.

A word of thanks goes out to all individuals that took part in the live tests. Moreover, we would like to especially thank Jan-Willem Manenschijn from Popup Escape for all his efforts and guidance during the project and Willem-Paul Brinkman for his ample advice and insightful comments.

# Summary

Escape rooms are an emerging market. In these physical and mostly group based games, players are locked in a room with the goal to solve puzzles within a certain time limit in order to 'escape' the room. To make sure players have a pleasurable experience, they are monitored during the game by a so called game master. This person helps players during the game. When players get stuck the game master gives players a hint. Another possible task of the game master is to create an immersive story within the escape room. The game master can do this by communicating with the players while taking on the role of certain characters that are part of that story.

The goal of this project was to develop the **Escape-Master system**: a system that automates the role of the game master. In the system the game master can define an arbitrary escape room by inputting puzzles, characters and dialog for those characters. This dialog can include hints or storyline narrative. With this information provided, the system can automatically interact with the players. The system keeps track of the progress of the players and uses this information to interpret questions they provide to the system using natural language processing. After this the system can provide a fitting response, which can be a hint or narrative that progresses the story of the escape room. The system can also start interactions with players on its own, based on time passed of their progress in the game.

The project consisted of two phases: research and development. During the research phase information was gathered on player interaction, user engagement, and chatbots. During the development phase a system was constructed that permits the game master automate the process of giving hints and create an immersive story. Additionally, an interface was build to give the game master the ability to define an escape room and start games while being notified of the actions the system takes. It was tested in a live setting on two occasions, which yielded favorable results in terms of user engagement and immersion.

# 1 Introduction and research

## 1.1. Introduction

Popup-Escape is a company that creates and hosts escape rooms. Their escape rooms can be played on a location indicated by the customer or in Delft. The escape rooms are highly customizable to deliver a personalized experience. The escape room called 'Social Hack' is the main subject of this project. The Social Hack involves competing with other teams to solve a number of interconnected puzzles. Most of the puzzles are presented in a an online environment via a website. The story accompanying the puzzles is delivered via text documents, emails, phone messages and theater. The use of multiple media creates an immersive narrative. A game master from Popup-Escape monitors the progress of the teams and provides hints and information when necessary. Currently, the game master needs to be trained in order to be able to host the escape room. Also the number of teams that can compete needs to be limited in order to realize a satisfactory escape room adventure.

**The first goal** of the project Escape-Master is to automate the role of the game master and tell an exciting story using a chatbot. Chatbots can interact with players via multiple channels like SMS and email to give the hints and information previously given by the game master. The online nature of the Social Hack provides information about the progress of the players which complements the context sensitive responses of the chatbot. Using not only the messages send by players, but also the data gathered from the puzzle websites, the chatbot should be able to engross players in the story of the escape room.

**The second goal** is slightly more ambitious: the modularization of escape rooms similar to Social Hack. By separating the system from the actual content of the escape room, it becomes possible to use the system for every escape room that has a similar structure. An escape room like the Social Hack consists mostly of puzzles and a small amount of story. It can be seen as a graph structure of puzzles. With an interface to construct this graph, the development of new escape rooms becomes quite simple. Reusability and replayability are two major positive consequences of modularizing.

## 1.2. Problem definition and analysis

As part of the research, the problem needs to be defined in more depth so there is a complete view of the problem that needs to be solved. Furthermore the problem needs to be analyzed to break the problem into smaller sub-problems and get an idea of the components the system needs to have to solve the problem.

### 1.2.1. Definition

Popup-Escape creates personalized escape rooms for companies to create the most fun and special experience for the client. The company has a new escape room concept where multiple teams of approximately five people will all play the same escape room at the same time, competing against each other. The puzzles of this escape room are mostly digital, which gives the possibility to digitally track the progress of the teams. Each team tries to solve puzzles while inside an escape room, but teams often are not able to solve a specific problem, which means they are not able to progress in the game. **The game master** is responsible to assist teams whenever this occurs, however this becomes a difficult task when a lot of teams are playing at the same time. This means the game master is limited to a small number of teams, hence Popup-Escape wants to have

an **automated game master (AGM)**, which was implemented in this project as the **Escape-Master system**. The Escape-Master sytem is a system that is able to automate the role of a game master in order to support a larger amount of teams. This allows Popup-Escape to host more teams at once.

### 1.2.2. Analysis

The AGM can be analyzed better by using a structure that consists of initiatives, epics and stories. Initiatives are a set of epics that all share a common goal and epics are tasks that can be broken down into smaller tasks, and those smaller tasks are the stories. During the project the client discussed four major initiatives of AGM. Those initiatives are: automation of hints, modular architecture, interaction with creator and communication with players. All four initiatives need to be realized in order to realize AGM.

The first initiative is automation of hints. The AGM needs to automate hints when teams are stuck during a puzzle. In order for hints automation to work, it is necessary to detect when players need a hint. Also, the game state needs to be retrieved so that a hint is given related to the puzzle at which a team is stuck at.

The second initiative is modular architecture. This is another important initiative of AGM because every escape room is different and AGM needs to be as generally applicable as possible. Modular architecture can be broken down into dataset, model characters, model puzzles and model users. Dataset is about all data that is related to an escape room that needs to be saved. Model characters are the characters that are part of the story of the escape room. Model puzzles are the puzzles of the escape room. Model users is about specific data from the users themselves (e.g. email, phone number etc.). These four epics need to be realized in order to make a modular architecture.

Interaction with the creator is the third initiative. The creator of the escape room can interact with AGM to define and adjust an escape room, as well as host the escape room during runtime. This can be broken down into the **Human Fallback System (HFS)** and **Creator Input System** (CIS). The HFS is a system in which the game master can be notified when the AGM is not able to generate a response during gameplay, which allows the game master to intervene. The CIS is a system in which the creator can input new escape rooms.

The fourth and final initiative is communication with players. This is about AGM communicating with players by using a chat bot. This initiative is broken down into the epics: natural language and engagement. Each team can utilize the chat bot and start a text based chat with a chat bot that creates a natural conversation with a team. The chat bot also needs to provide an immersive experience for the teams in such way that the teams are comfortable with the conversations and that the system plays an active role within the story of the escape room.

From the mentioned initiatives and epics, a roadmap is constructed for the project. This roadmap is a clear overview that breaks down the big problem into four initiatives, and from each initiative it is again broken down into epics. For information on the project timeline, the Project Plan can be read in appendix H.

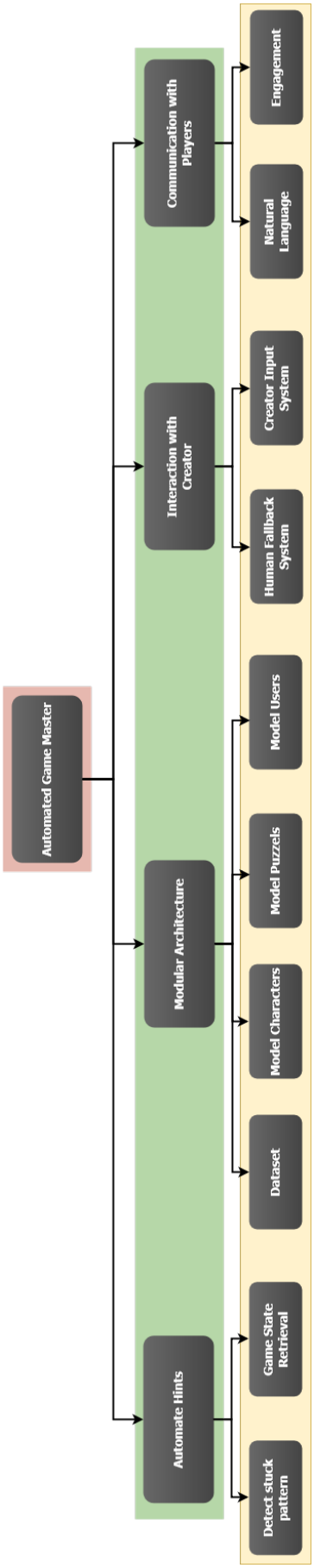



Figure 1.1: Roadmap of the project. Initiatives in green. Epics in yellow.

### 1.2.3. Gameplay

The escape room that the end product was tested on is called **The Social Hack**. The gameplay of The Social Hack is as follows. The game master starts with a short introduction about the goal of the escape room. The game master does this while playing the role of a character within the story of the escape room, e.g. an agent of the Dutch Intelligence Service. After the introduction, the game master will leave the room and start the timer. The teams will then start with solving puzzles to reach the end goal.

Every puzzle is connected to another puzzle, and there are multiple ways to reach the end goal. Each puzzle may consist of a combination of physical and digital puzzles. An example of physical puzzle is a wooden box with hidden compartments. An example of digital puzzle is a website where the user has to crack a password using social engineering.

Each team has at least one laptop with internet connection that is used to solve digital puzzles but it can also be used to send emails with questions to characters within the story. The game ends when the timer has elapsed or when the end goal has been reached.

## 1.3. Product goals

### 1.3.1. Interaction Design

To ensure that the product is well developed based on the needs of the target audience, user centered design methods have been used. This will result in a more effective product designed specifically for user experience.

For this product there are two different users. The game master, which leads the escape room, and the players that participate in the escape room. The target audience is defined in the next section.

Target audience

The product's target audience is divided into two types. The players and the game master. The players are people that participate in an escape room and solve the presented puzzles. These players usually work together in a group to accomplish this. These players are employed adults with higher than average affinity with technology. Also these players are usually higher educated and analytic thinkers. The game master role is taken on by employees of Popup-Escape, who are usually university students.

Conceptual phase

During the conceptual phase certain methods are used to create a clear idea of the users and usage of the end product. Two different techniques used for this are personas and scenarios. This phase is an important stage of the design process to customize the product to suit precise demands [7]. The personas can be seen in figure 1.3 and 1.2 and the scenarios can be seen in figure 1.4.

**Personas** Two types of personas have been created based on the target audience. By creating personas based on the target user group, the goal to ensure a good user experience can be achieved [15].

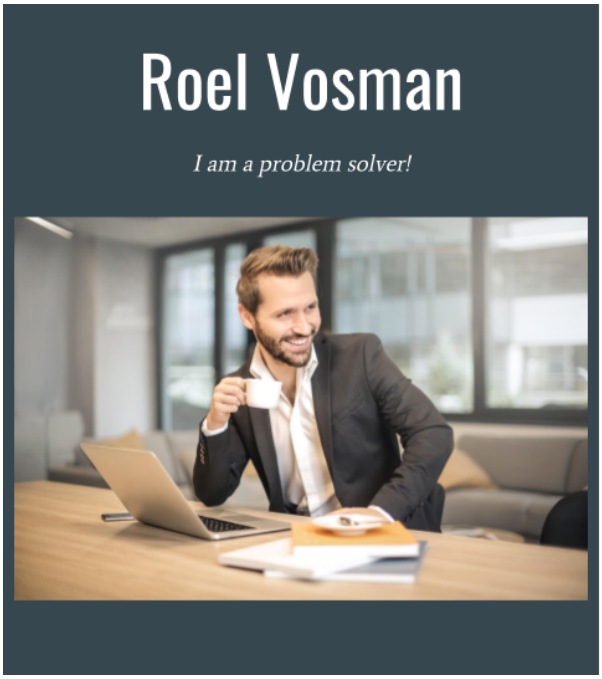
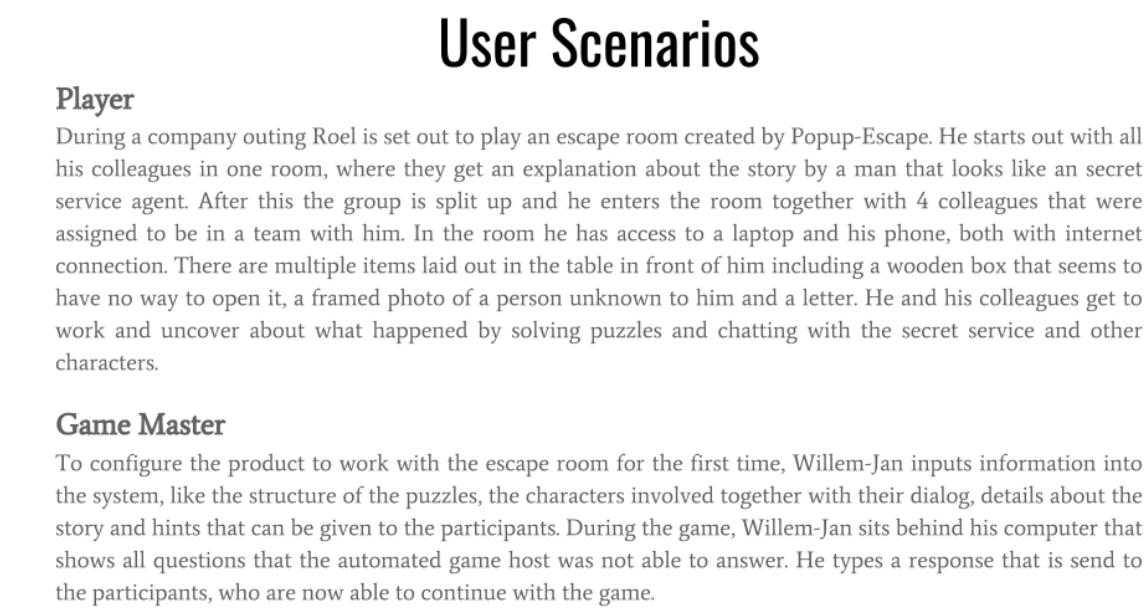



Figure 1.2: Persona of the player.


Figure 1.3: Persona of the game master.

**Scenarios** By creating a user scenario, the basic story of an action or goal that a user wants to accomplish is described [8].


User Scenarios

Player

During a company outing Roel is set out to play an escape room created by Popup-Escape. He starts out with all his colleagues in one room, where they get an explanation about the story by a man that looks like an secret service agent. After this the group is split up and he enters the room together with 4 colleagues that were assigned to be in a team with him. In the room he has access to a laptop and his phone, both with internet connection. There are multiple items laid out in the table in front of him including a wooden box that seems to have no way to open it, a framed photo of a person unknown to him and a letter. He and his colleagues get to work and uncover about what happened by solving puzzles and chatting with the secret service and other characters.

Game Master

To configure the product to work with the escape room for the first time, Willem-Jan inputs information into the system, like the structure of the puzzles, the characters involved together with their dialog, details about the story and hints that can be given to the participants. During the game, Willem-Jan sits behind his computer that shows all questions that the automated game host was not able to answer. He types a response that is send to the participants, who are now able to continue with the game.

Figure 1.4: Scenarios of the game master and player.

### 1.3.2. Requirements

The functional and non-functional requirements are based on the Roadmap and the interaction design. By defining Initiatives and corresponding Epics, certain tasks can be found per Epic. These epics and tasks lead to functional requirements.

The most important functionalities of this project are the ones that allow escape rooms to be hosted easily. In this case, the creator of the escape room should be able to construct the necessary puzzle and story elements and the state of the game should be recorded. These 'input' functionalities should result in events that trigger the sending of messages to the players. Of less importance are the extensions of this basic product: communication via other media and special types of events.

Table 1.1: The project requirements defined with MosCow.

| | Functional Requirements |
|---|---|
| Must haves | Ability to support multiple teams at the same time |
| | Detect when players need a hint, and communicate these hints using dialog by a fictional character through a text based medium, like chat or email |
| | Possibility for the creator to input an escape room (order of the puzzles, hints, story, etc) into the system |
| | Possibility for the creator to create fictional characters with unique dialog |
| | Natural language interaction in English with fictional characters, by the chatbot |
| | Retrieve and use game information (e.g. answers to puzzles, number of answer attempts) from existing Popup-Escape system |
| | Ability to reconstruct the current state of the game |
| | Ability to know which messages are sent by which team |
| Should haves | Interaction with players via more than one medium (e.g. chat client, e-mail and social media) |
| | Ability to hold basic user information (e.g. email, telephone number) to link back the different forms of communication to the right team |
| | Human fallback system that makes it possible for the game master to answer questions that the AGM does not have an answer to |
| | User friendly UI based on expert review |
| | Pro-active events based on the storyline and game progression |
| Could haves | Interaction with players through other types of media, like speech and video |
| | Data analysis and machine learning from previously collected data from escape room teams to automatically improve performance |
| | Automatic adjustment of escape rooms based on game progression (e.g. giving a team harder puzzles if they solve easier ones really fast) |
| | Different escape room story endings based on performance |
| | Scoreboard that shows progression and score of teams |
| | Model individual players (e.g. their knowledge base) |
| Won't haves | Game master communication with the chatbot |
| | Simulating games with multiple teams |

Non-Functional requirements:

- The system will have a test coverage of 70%
- The system will be user tested by the target audience in at least two occasions
- The system is extendable by Popup-Escape
- The code quality is checked by SIG
- A code analysis tool is used throughout the project
- The system will make use of continuous integration, Travis CI
- The system will be ethically evaluated
- Interface should be accessible online from every computer
- The system should be scalable up to at least 100 groups of 5 people
- The application should run in a server environment

## 1.4. Groundwork

There are four different Initiatives in the Roadmap which are the main components or the system. Automation of hints, modular architecture, interaction with the creator and communication with players.

### 1.4.1. Automation of hints

One of the main tasks of the game master is to help out players that are stuck by giving hints. Players only have a limited amount of time to play the escape room, and being stuck on one particular puzzle for too long will have a negative impact on the player's experience. On the other hand, giving out hints too soon will take away the users enjoyment from solving the puzzles. The final product will have to find a way to detect the right moment to give out a hint. This section will be devoted to research on how this automation of hints can be achieved.

**Game state retrieval** In order to give a hint to the players it is important to know what puzzle the players are working on. Certain information is available from the system that is currently in use to host the escape rooms. This information includes:

- Answers given by the players
- Number of attempts for answering a puzzle
- Tracking the access to websites that are part of certain puzzles

If this information is combined with information about possible paths through the escape room, it is to know what puzzles the players currently have access to. It is good to note that there are sometimes multiple paths to the final solution, as well as physical puzzles. Therefore it is not always possible for the AGM to know which specific puzzle the players are working on without extra input from the players. When needed, the AGM could ask the player for this information in the form of a natural language interaction.

**Detecting when players need hints** One of the main questions that will need to be solved in order to automate hints, is the detecting when players need hints using the data that is available to the system. To do this it is important to consider the way users play escape rooms. For example, there is data available about the number of wrong answers given to a puzzle by the players. You could consider a large number of wrong answers as a sign of the player needing a hint. However, by experience the Client told the team that there are players that will just try to 'brute force' their way through a puzzle by trying a lot of different answers, without first searching for the right solution. Giving a hint after a large number of wrong answers is therefore not a good indication of a player needing a hint.

This shows that it is especially important to test solutions on real players and implement their feedback in an iterative process. This is the approach the team will take. The team will start out with an educated guess that seems reasonable and has scheduled user tests to test the solution. At the end of these tests the players will take a survey on the product, including questions about the timing of the hints. Did they get a hint while they didn't need one and vice versa? These surveys will be analyzed together with other game data to search for improvements in the current use of the data.

**Giving hints in the form of an assistant** Lieberman [18] defines an agent 'as any program that can be considered by the user to be acting as an assistant or helper, rather than as a tool in the manner of a conventional direct-manipulation interface.' An important observation is that if users perceive the agent's actions as actions that they could have done themselves, they are more willing to conceptualize the agent in the role of an assistant. This fact could be used to give hints in a way that makes the fictional characters feel more like assistants. If, for example, the system predicts that the users are stuck on a puzzle that involves opening a box, fictional character could ask *'would you like my team to find out how to open the box?'* (an action that the user could have done with his own team) instead of *'do you need a hint with the current puzzle?'*.

**Machine learning to improve timing of hints** As seen in Section 1.3.2 the team is considering using machine learning to improve the timing of hints. There are multiple things to consider when choosing an algorithm for this problem. First thing to note is that this is a classification problem with two classes: giving a hint and not giving a hint. Secondly, it has to be noted that there would only be a small amount of data available to train the classifiers. Thirdly, the feature vectors would only have a small amount of features in them, which makes algorithms like SVM less applicable [5]. Finally, since this feature does not have high priority in the project, the classifier will have to be quick to implement and train. Using an overview on machine learning provided by Microsoft [5], we can identify that the decision tree, the perceptron and two class logistic regression are suitable algorithms to solve this problem.

### 1.4.2. Modular Architecture

Designing an escape room is a creative process. They are created, tested on real people and adjustments are made where needed. The final product should be able to incorporate these changes quickly and generate a new automated game master. The intent is to make the system as generally applicable as possible. It should work for any arbitrary escape room, as long as the the creators provide content for it. This raises multiple questions. What elements are needed to model an arbitrary escape room? What content should the creator provide in order to create a pleasurable experience for the players? This section will contain research into what escape rooms exactly are and how they are played and created in order to find a general model for an escape room.

**State and future of escape rooms** The escape room concept appeals to a multiple of demographic groups. To be able to customize the experience would enhance the players engagement. Players should be able to set constraints, similar to in a video game. Some escape rooms, like Popup-Escape, focus on letting teams compete. There is a central leader board which tracks the progress of all teams. This recording of advancement tries to supplement the pressure from the time limit. Some groups will like this aspect of the escape room experience, but others will opt for a more casual experience. Nicholson [21] identifies three principal group types: casual, standard, and hardcore. For each group a disparate quantity of hints, solution, and assistance should be provided to meet their desired experience.

An escape room is often played once and can not be replayed, because the answers to puzzles and important revelations are known from playing the previous time. This pay once business model also holds for Popup-Escape. Players pay once and return only when a new escape room is introduced. The video game industry can again be used as an example. Most contemporary escape rooms have many similarities with point-and-click adventure games: locating items, combining items, and solving puzzles. The puzzles and revelations in both cases static. To produce a replayable experience the focus should lie more on the adventure than on solving the static puzzles. Players should be made part of a spectacle, feel heroic, and engage with the presented challenge. To implement the replayable experience there are a few options; a different experience is presented for every playthrough. These options include:

- Computer-generated puzzles
- Pick a random subset of a large set of puzzles
- Generate a random path through a graph of puzzles (=nodes)
- Group-based puzzles
- Set of good and bad endings
- Set of various good endings

Replayability makes it harder for players to cheat to obtain a better score. More important from a business perspective is the amelioration of return costumers and consequentially of revenue. Viewing escape rooms as live-action adventures where players are placed into a role could be the future. All puzzles in such an adventure should contribute to the narrative, entertain and captivate, provide an a-ha moment, and encourage group interaction.

**Modularization** Dividing a system into smaller parts is called 'modularization'. The practice of modularization results in a codebase that is more manageable, flexible, and comprehensible [23]. This leads to a shorter development, demonstrated by the following examples. When an error occurs the corresponding module needs to be debugged, not the whole system. If new features have to be added a new module may be created or an existing module may be extended. Each module has its own, predefined task, which causes a clear connection between the code and its purpose. If one wants to quickly know what the function of a module is, one only has to look at the interfaces of the modules. As the interfaces specify the input and output of a module, it can be seen as a black box, abstracting away its inner workings.

Modularization can be applied to escape rooms. The modules and their relations form a hierarchical structure graph. The modules on the first level don't depend on any other modules. No cycles should exist in the structure, as this would mean that modules should be combined into one bigger module. Drawing this graph while discussing the design decisions helps optimize the modularization process.

Modularization maps all design decision and is created through discussion by the programmers involved in the project. The decomposition can be done in various ways. One option is to make the 'major steps' of the process into modules. A flowchart can be constructed that describes the process, where each node would represent a module. Another option is to make each module based on information hiding. The goal of this decomposition is to hide the implementation of functionalities from other modules as much as possible. This option makes the most use of interfaces. The parts of the system that are most likely to be modified in the future should be put in a separate module. Thus, the rest of the system doesn't suffer from regression. The second option is most often the best choice, because during the development process design choices will change. It becomes hard to maintain flexibility when employing the modules based on process steps.

**Database** Based on the modularization one can construct a database containing information of escape room games. Four main entities are considered: Teams, Puzzles, Characters, Game.

- *Teams* represent the group of players participating in the escape room. They will have a number of members, a name, and related channels (Whatsapp, Facebook, etc.).
- *Puzzles* are the challenge that the escape room represents. They will have question statements, descriptions, answers, hints, and difficulties.
- *Characters* will represent fictional persona's enacted by chatbots. They will have names, dialog options, background information, and profile pictures.
- *Game* holds the general information of an escape room game. It will have a global time, time limit, number of teams and other general relevant information.

The main decision when choosing a database is relational and non-relational. Since the chatbot will receive data supplied from online channels, it will most likely receive JSON formatted objects. Non-relational database, like MongoDB [6], use this format natively. Therefore it would be best to opt for the flexible and complementary nature of the non-relational database.

Implementing a database can be made easy with an Object Relational Mapper (ORM) or Object Document Mapper (ODM). No query language has to be used. The development can be performed in the programming language of of the entire project. This comes at the price of slower queries. As this project only retrieves small quantities of data for single games, the added query time is negligible.

### 1.4.3. Interaction with Creator

Besides the players of the escape rooms, another important user of our system are the creators of the escape rooms. They will need a system where they can define the specifics of the escape room, identified in 1.4.2, so the system can generate an automated game master. On top of that, a helpful feature would be a human fallback system that gives the game master a way to monitor the state of the different escape rooms and answer questions that the automated game master has trouble processing. This section will discuss how such systems could be created.

**Human fallback system** Chatbot systems are typically adapted to handle simple requests of users and escalate to human support when the request get more difficult [24]. This system will be no exception. When the chatbot fails to identify a proper response it should hand over the task of responding to a human by the means of a fallback system. This system will relay the input to the game master. He or she will then respond with an appropriate answer. It should eliminate non-sequiturs and disguise the fact that the player is now talking to a human instead of a machine. The fallback system could also issue a message to the game master when players are taking a long time to complete a puzzle and the hints given by the chatbot don't seem to be helping.

**Escape room input system** The escape room input system requires a more detailed description of the escape room. Therefore it would not be feasible to implement this as a chat dialog. To keep the whole process contained within the chat environment, the chatbot can redirect the creator of an escape room to an interface located elsewhere. When all the information for the escape room is complete and submitted by the game master the chatbot will confirm that the creation of the escape room has been successful.

User interface (UI) design is a field on its own. When designing an UI the principal problems that arise are: *"... getting users' requirements, writing help text, achieving consistency, learning how to use the tools, getting acceptable performance, and communicating among various parts of the program."* [20].

The use of tools speeds up the implementation process significantly, these include: window systems, toolkits, interface builders, and user interface management systems. It is important to note that for different SDK's different tools are available. For this project the SDK would depend on the chatbot framework, so deciding which tools to use would have to be performed based on this decision.

### 1.4.4. Communication with Players

In order to supervise players of the escape room, it is needed to establish a line of communication with the players. This line of communication will be used to provide the players with hints and new information they need to solve puzzles. The other way around it will be used to give players the option to ask for hints and talk to the fictional characters to advance the storyline of the escape room. The main requirements for the communication is that it is natural and immersive. This section will be devoted to research on how to achieve these goals.

**Human-like behavior of chatbots** Imitating human intelligence and behavior has been a challenge for computers since the idea of the Imitation Game, or Turing Test was first proposed [25]. The Loebner Prize Competition is the first formal instantiation of the Turing Test [14]. The chatbots in this competition use technical and language approaches to fool the judge in believing they are human. Although none of the winners were able to pass the Turing Test, they did epitomize the most human like behavior at that time.

Pattern matching is the most common approach from the technical approaches. Input is simply matched with a pattern and the corresponding response is sent back to the user. Parsing is similar to pattern matching; this method reduces input to a set of words. Modern chatbots can parse the complete grammar of natural language sentences. Another method is that of the Markov Chain Model. The probability of the appearance of a letter or word in a textual context is fixed. The Markov Chain Model uses this idea and predicts the occurrence of a letter or word based on previous input. The order of a Markov Chain Model specifies the amount of consecutive occurrences taken into account. A more convoluted approach is that of an ontology or semantic network: a set of concepts that are connected in a hierarchical or relational fashion. Computers can represent ontologies with graphs. Since there are various algorithms to determine characteristics of a graph, a knowledge base can be created from an ontology. Recently relational databases together with SQL have been employed to make chatbots 'remember' previous conversations [12]. Combined with the parsing this can be a powerful tool. An input from the user can be reduced to a simple query relating to a certain topic present in the database. A more meaningful response can be created from older inputs.

Two chief languages were used in the competition: AIML and its successor ChatScript. AIML is a language used to describe input rules and analogous responses with an XML-like syntax (see Figure 1.5). One of the major strengths of this language is that it is recursive, so that input rules can be passed on to other input rules. ChatScript, has an easier syntax and adds a number of functionalities that make ontologies an inherent part of the language. When no input matches transpire from the AIML input rules, ChatScript tries to create a reasonable default answer.

```
<category>
<pattern>HELLO</pattern>
<template>Hi, how are you?</template>
</category>
```

Figure 1.5: Example of a simple AIML input rule

The language approaches use general understanding about conversations to make the chatbot more convincing. An easy trick is to simulate typing errors or to make it appear as if the chatbot is typing, taking time to formulate its answer. This will make the behavior of the chatbot seem increasingly human. Additionally, the chatbot can be given a personal history. The responses include stories on childhood, parents, job, etc. When no appropriate match from the technical approaches is found, a set of hard coded responses can be employed. These so called 'canned response' are the same as very specific patterns. A chatbot could be build using only canned responses, but this would not be practical.

**Player engagement** User engagement is a term that is used to describe the experience of users that use technology. User engagement plays an important role in the escape room, because users need to be engaged in the story and the puzzles. The product that needs to be delivered needs to have basic functions for the escape room but also need to provide user engagement. If users are not engaged, people will have a bad experience and will not return to the escape room at another date. People might go elsewhere that does provide user engagement. One way of keeping user engaged is to prevent players from getting stuck with a puzzle. Questions that may arise are for example: how to determine that players are stuck with a puzzle by using chat messages? Or how to determine that players are stuck with a puzzle by using the game state? In order to get a better feel for these kind of problems, it is needed to understand and give a better definition to the term 'user engagement'. The goal of this research is to look at key components of user engagement to help with creating the right user engagement for the escape room. In order to achieve this, existing research about user engagement was reviewed.

A model of user engagement consists of four stages of engagement: point of engagement, engagement, disengagement and re-engagement [22]. The point of engagement is about aesthetics or motivation of the user that might capture the interest of the user which starts the process. After the initial point of engagement, actual engagement is started. At this point in time it is about maintaining the interest of the user and getting responsive feedback from the application that is being used. After engagement is dis-engagement. When users enter this stage there can be many reasons that can be either good or bad. Users might disengage because a puzzle is successfully solved which creates a positive emotion or disengage when a user is stuck with a puzzle which creates a negative emotion. After dis-engagement users may start re-engagement which is to re-enter the point of engagement to start the process again. While playing the escape room, users might restart this process multiple times during a single puzzle.

Also an important part of engagement is considering a users reaction to a certain interface. For example, an interesting finding was that users with different cultural backgrounds have different responses to the same interface [17].

**Chatbot frameworks** There exist a multitude of chatbot frameworks. When deciding what framework to use it is most important to take into account what features are provided, but licensing, channels, language support, documentation and technical details should not be disregarded. In 2017, Mindbowser conducted a survey on chatbots with participants from nine different industries. In this section the most popular choices will be enumerated and compared, listing the various aspects relevant to the development process [19].

The first choice of framework for building a chatbot was IBM Watson [16]. Watson Assistant, formerly know as Conversation, embodies three components: intents, entities, and dialog [11]. An online workspace is provided where the chatbot can be extended and data analytics on the conversations are available. The correlating API supports the Node.js, Python, and Java SDK's. There are three pricing plans: Lite, Standard, and Premium. It is clear why IBM Watson is the first choice: the online workspace supplies an easy interface, extending your chatbot can be made as hard as you want with simple intents or with extra middleware from the SDK's. The documentation of the framework is extensive and helps you set up a project quickly. The largest flaw is the cost, if the chatbot receives a lot of traffic it will become increasingly expensive.

The second choice was Wit.ai [9]. It is a free framework and has a Node.js, Python, Ruby and HTTP API client. Other unofficial clients are available on github. Around 50 languages are supported. Wit.ai uses entities, intents, and actions to respond to users. Natural language processing is used to understand the meaning of all input and extract relevant information. The documentation of Wit.ai is especially useful, as it contains recipes that let you implement the functionality that you want using a step by step guide [10]. Facebook integration is supported natively. To use the bot in an application one of the clients can be used, integration with a website is now only possible via JSONP.

The third choice was the Microsoft Bot Framework [1]. Their Bot Builder SDK provides the tools needed to develop a bot and their Bot Framework lets you connect bots to channels. An online environment is provided to create and manage bots, which makes writing code optional. Although, there is support for the .NET en Node.js SDK's for fabricating more complicated functionality. What makes Microsoft Bot Service appealing is: bot templates, multiple language support, and the flexible deployment of bots on different channels. Other Microsoft services can be integrated for added functionality (speech, search, voice). Microsoft Bot Framework is open source and available on GitHub. However, connecting your bot to your own website will cost money.

The TU Delft Supervisor for this project suggested Google DialogFlow [2]. DialogFlow comes in a Free and Enterprise edition. The Free edition allows unlimited text queries, thus should be sufficient for the Popup-Escape use case. An online environment gives developers the tools needed to create entities, intents, view

analytics, and train the chatbot. An inline editor and webhook can be used, but a Firebase subscription is needed. Integration for an array of platforms is supported, to list a few: Slack, Facebook Messenger, Telegram. DialogFlow has three API components, based on use cases: Fulfillment, Detect Intent API, Agent API. Fulfillment integrates the natural language capabilities into your own website or services. Detect Intent integrates the whole DialogFlow interface into a website. The agent allows you to dynamically change the agents behavior. Fulfillment support only an Node.js SDK, but can be used by any platform that works with HTTP requests. The other two components are supported by a large number of SDK's. These client libraries can simply be forked on GitHub. The documentation is simple, yet has all the necessary information.

# 2 Design

The design of the product is split into two canonical parts: front-end and back-end. The front-end includes the interaction with players and creator. The back-end is made up of a model of an escape room and the natural language processing functionality. The part of the front-end that interacts with players is simply related to the channels used to communicate. The part that interacts with the creator should have an interface that helps them in the design process of the escape room.

## 2.1. Initial designs

The chief influence on our design is the choice of chatbot framework. Frameworks support different languages and thus application programming interfaces (API). The IBM Watson framework was chosen for the reason that it supports multiple languages (node.js, Java, Python), has sufficient documentation with examples, and was elected as the number one choice for creating chatbots in a large survey of developers.

Because the functionality of the Social Hack is hosted on a Meteor server, the use of node.js would integrate easily with the software already present. Objects in JavaScript are analogue to their JSON representation. Creating objects through requests is made trivial. However, the team's knowledge base for the Java API was greater. Java's Spring Boot provides the same functionality as could be found in JavaScript. Processing JSON objects proved no problem, because of libraries like Gson. The size of the project was a convincing factor in choosing a language that was better known. Testing experience and the ability to use a familiar and an easy-to-use IDE were the final elements that made Java the preferred alternative.

IBM Watson Assitant API was extensively scrutinized. How would the natural processing functionality be integrated into the system? Can different media be used for communication? Watson has sufficient intelligence to deliver the simple dialog needed for hints. By supplying only a limited number of example sentences Watson was able to deduce the right intent. The focus was shifted from selecting a chatbot framework to the creation of an environment for the creator of the escape room.

Spring was elected to create this interface since it makes it very easy to create web applications in Java. The Model-view-controller (MVC) prototype is used to create a maintainable and readable code base.

The project was split into two parts analogous to front- and back-end: Spring and Watson. Using the MVC model, the forms submitted in the Spring interface are stored into objects, which Watson then uses to create a chatbot. This model is a vital part of the whole Escape-Master system, see figure 2.1.

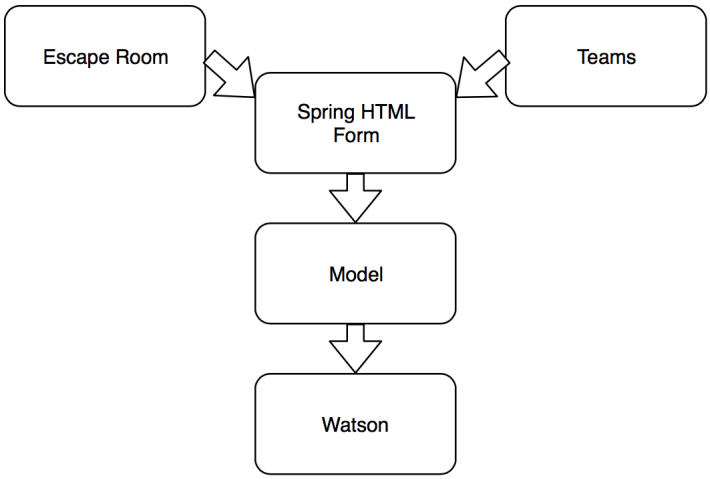



Figure 2.1: Abstract diagram of Initial Design

## 2.2. Challenges

When the Social Hack escape room is played, multiple events occur. Players answer and solve puzzles, access websites, and get stuck (i.e. are unable to solve a puzzle and progress). These events need to be monitored in order to get the right context for the chatbot. Fortunately, Popup-Escape already had a system in place. There needed to be a way to process the information from the Popup-Escape system. Although this wasn't very complex, the proper interpretation of the data is quite difficult.

The product would consist of multiple components: the chatbot, the communication channels, the interface for creating an escape room, and the interface for monitoring an escape room. Integrating these parts requires precision and common practices. The relevant knowledge needs to be located in the right objects.

The short solutions to these challenges will also be presented here. In chapter 3 a more complete description and implementation of the solutions to the challenges can be found.

### 2.2.1. Watson

The challenge is combining the Watson API with other components of the product. One example is combining the component puzzles with Watson. An Escape-Room has a lot of puzzles, and people may have trouble solving a specific puzzle. Thus, questions can be asked by people about their current puzzle. This means that Watson needs to give hints about the right puzzle instead of giving a hint about a puzzle that is already solved. So changes within a game state must change the behavior of the hints that Watson gives. This problem was solved by having regular meetings with every team member. In these meetings every component was sketched on a whiteboard and also how each component interacts with each other.

### 2.2.2. Messaging

Interaction with players is done by using messages. Communication for this project incorporates two channels: SMS and email. SMS was fairly easy to use, however email was a challenge in this project. There were two main challenges for email messaging. The first challenge is understanding how to perform actions for email within Java. Many subjects play a role here, an example is user authentication. User authentication seemed to differ depending on the email service that is being used. For example, Gmail from Google requires developers to use OAuth 2.0 protocol, in order to gain access to Google's email API. Moreover, an access token is needed in order to use the API. This problem was solved by discussing the email messaging with the client. Luckily, Popup-Escape is very flexible and is able to provide their own SMTP server so we did not have to deal with this problem. The second and biggest challenge was reading email from within the inbox. The message content could not be easily received. The content was often not stored as plain text and was instead stored as variety of types. Each type needed a specific parser in order to read the content, furthermore some types were "part" types and had to be combined in order to receive the full message. Sadly the parsers were not provided within the API that was being used in this project. So this problem was solved by writing the parsers manually and researching online where similar parsing was needed.

### 2.2.3. Game states

When playing the Social Hack or any other escape room the state of the game changes constantly. The principal change is caused by solving puzzles. Each game is timed and terminates after the time limit is reached. Therefore, a time factor has to be taken into account as well. The Social Hack is played by multiple teams. It is imperative that each team has a corresponding state. The information for each team is gathered by the system created by Popup-Escape for the Social Hack. This makes it so that the Escape-Master system only has to process the data. The principal issue was that not all puzzles of the Social Hack are digital, for these tracking the state is not feasible. Conversely, for the digital puzzles, which consisted mostly of password submissions and site accesses, the solving can be tracked quite easily. The state of the analog puzzles can be deduced from that of the digital ones.

The data per team is sent to an endpoint of the Escape-Master system by means of a standard HTTP POST request. The data is loaded into a Java object and the relevant data is extracted. The data is linked to an username in order to have the proper context for each team. The context is essential for supplying hints at the correct time. The publishing of hints is based on two types of events. Firstly, after a determined amount has elapsed since the start of the game, a team will be given a suggestion of where to start looking next. Secondly, if a determined amount of time has elapsed after solving a puzzle, i.e., a team is stuck, a hint relevant to the current puzzle is stated. The graph-like nature of the Social Hack makes it uncomplicated to ascertain the current puzzle.

### 2.2.4. Game events

An interesting functionality of the Escape-Master system are the game events. While creating an escape room the client can decide to input certain events. An event is a type of message that is sent by a chosen fictional character at a certain point during the game. For instance, the bad guy of the game can send an SMS to the players after completing one or all of the puzzles.

There are two different types of events:

- **GameTimeEvent** is an event where a message is sent a certain period after the game has stared. For instance, all teams receive an email after 5 minutes after the escaper room has started.
- **PuzzleTimeEvent** is an event where a message is sent after the completion of a puzzle or after a puzzle has been completed for a duration of $x$ amount of minutes. The latter can be used to give a team a hint for example, when a team has been playing a puzzle for 10 minutes the team can receive an SMS with a hint on how to solve the puzzle.

The implementation of these game events was an interesting challenge, as these events must be fired at the correct time and for the correct player. The challenge here was the different amount of possible implementations of these events and its logic, as this can be decided in multiple ways but deciding the best way was a process of trial and error. For example, the logic to decide the duration of an event can be done in multiple ways; start counting after a puzzle has been solved or when a puzzle has become accessible. These implementations all have different consequences on other aspects of the system.

Another challenge was the design of a suitable software pattern for the events. The events should be able to be easily extensible for later use when adding other types of events. Also an implementation must be designed to check whether an event should be fired for example, by using a timer or whenever there has been a state change.

## 2.3. Final Design

The initial design with Watson and Spring was further developed during the starting period of the project. The MVC-model fitted the design nicely. Despite that, an essential part of the system was overlooked in the initial design phase: the communication with players. When playing the Social Hack players get hints through two channels: SMS and email. The channels are not part of the creator input system or the interaction with Watson; it is the link between these two and the players. It was decided to implement this functionality in a separate package from the MVC-model. The two event types discussed previously, puzzle events and game events, are coupled with SMS and email. In this way the channels are abstracted away from the initial front- and back-end. The creator input system provides interface for creating these events, which then automatically trigger when a new Social Hack game is started and the game state is updated.

The complete system consists of three parts: Watson, Spring, and the communication channels. The creator only interacts with the input system created in Spring. The players don't see any interface, other than their own SMS and/or email client. Watson is not interacted with by anyone. The creation of the chatbot is entirely automated, see figure 2.2.

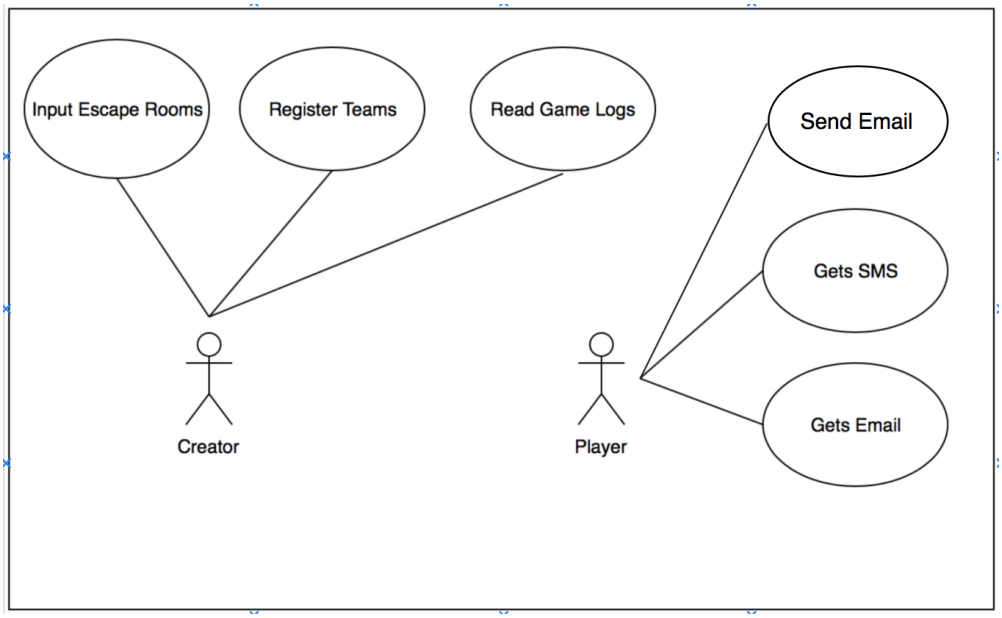



Figure 2.2: User diagram of the complete system

# 3

# Implementation

The system contains four modules, as explained in the Roadmap; the creator system, communication with players, automation of hints and modular architecture. The implementation of the system is separated into these four modules. A complete overview of the system can be seen in the UML, which can be found in appendix E.

## 3.1. Creator system

In order for the whole system to function, the creator of the escape room needs a way to input, start and stop the escape room. To do this, it was decided to use an online website with a graphical user interface. Since the escape rooms created by the client are mobile and can be set up everywhere, it made sense to also make the interface accessible from all computers with an internet browser. This way changes can be made on the go, without the need for special software to be present on the machine used to access the interface. The choice for a graphical user interface instead of a command line interface was made to make the application more intuitive and usable without the needing technical knowledge about the system. The goal was to make the GUI speak for itself, so everyone with knowledge about escape rooms could use it immediately.

### 3.1.1. Creating an escape room

Before we can play an escape room, it first needs to be defined in the system. Creating an escape room was split into five parts:

- Name: The name used to refer to this escape room
- Characters: The characters that this escape room uses to communicate with the players and the channel the character will use to do so (for example SMS or email)
- Puzzles: The puzzles in the escape room. Creating a puzzle can in turn be split up in two parts:
  - Name and puzzle id: The name of the puzzle and an identifier that is used to make sure any other system an escape room is using can update the state of puzzle in our system;
  - Dialog: For each puzzle, dialog for characters can be defined which consists of example sentences of what players may ask a character when trying to solve this puzzle followed by what the character should answer.
- Links: In this step the creator can define which puzzles are accessible after solving the puzzle, i.e. they are linked. This step can be seen as defining edges in a graph
- Events: Here the user can define dialog that characters should initiate with the players after a certain event happens in the game. There are two types of events that the user can define:
  - Game time events: These are events that trigger for all teams simultaneously, regardless of their progress in the game;

– Puzzle time events: These are events that trigger differently for each team depending on their progress in the game.

Since a lot of information is needed to define an escape room, the decision was made to split the process into multiple steps that each have their own page. This ensures that the process is not overwhelming the creator. All the screens shown during the creation process have a progress bar that indicates which steps are completed, and what the steps are still remaining. This gives the creator a clear overview of the process. Once all the steps have been completed the escape room can be saved to be started at a later time.

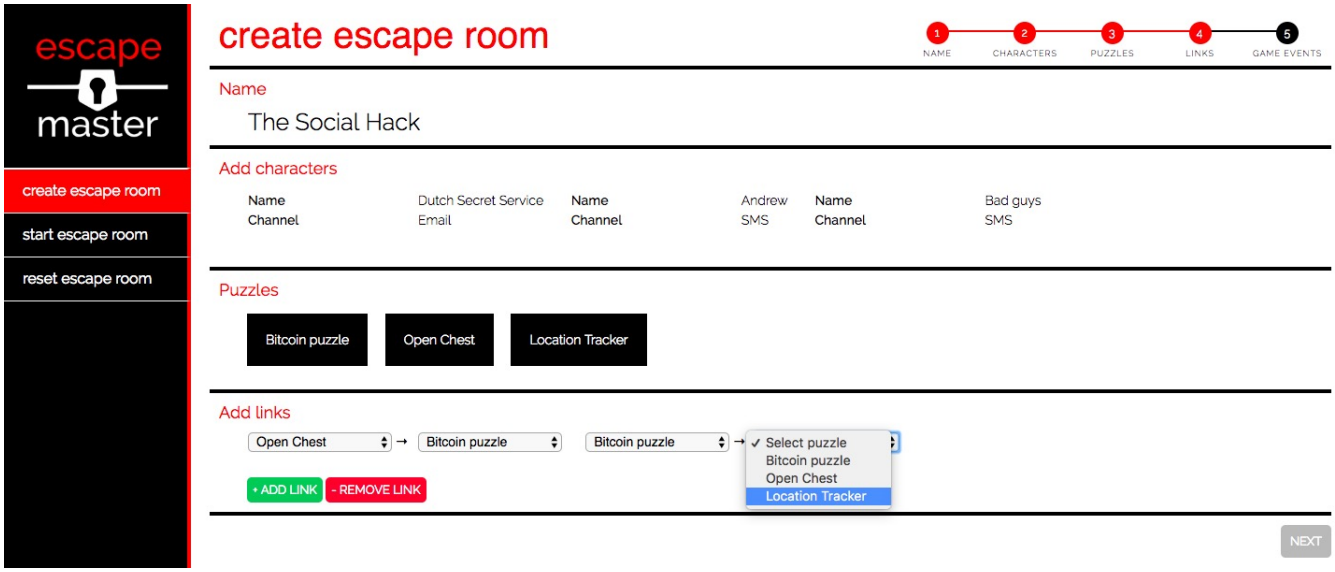



Figure 3.1: Creating an escape room in the editor

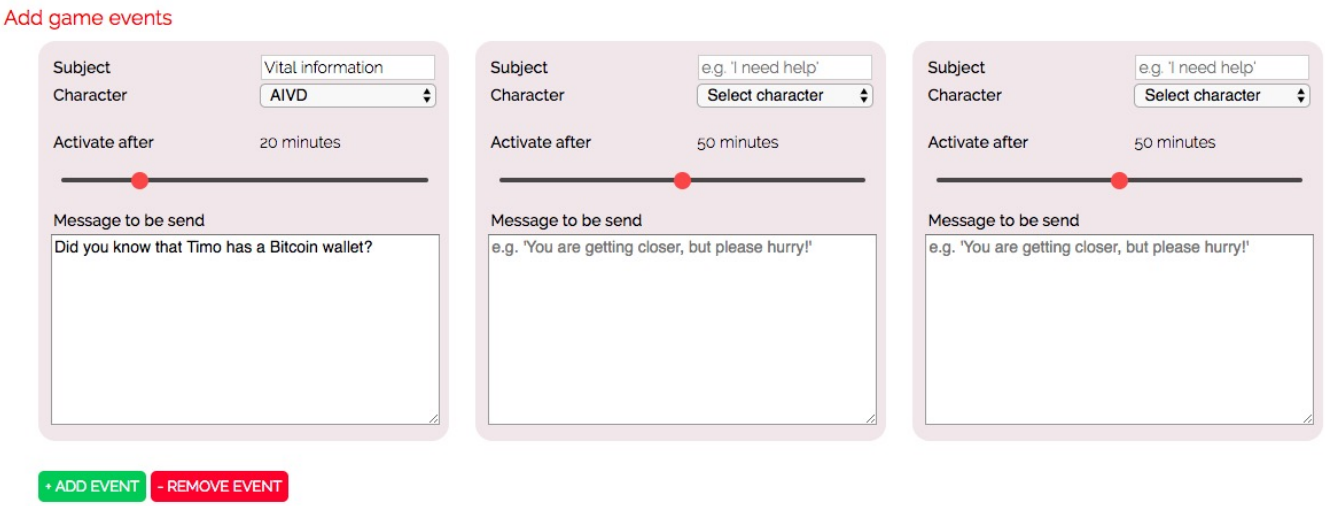



Figure 3.2: Fields for adding events

### 3.1.2. Starting an escape room

Once a escape room is created we can start it at any moment the creator wants. Starting an escape room is split up into two parts:

- Creating a game: The first thing to do when starting a game is creating the game. This can be done by selecting an escape room to play and inputting a CSV file containing the information of the players. This includes their team number, phone numbers and email addresses. This information will be used to contact the players during the game;

- Starting/stopping the game: After the game is created the only thing left to do is to start the game by pressing the start button. After the game is started a timer appears to show the creator how much time has elapsed since the game has been started. On this screen all relevant activity of the app is logged. This includes activity like sending and receiving SMS messages, receiving and answering emails and the status of the different teams. This helps the game master to verify that the automatic responses given by the program are correct and gives the game master the possibility to intervene if this is needed.


Figure 3.3: GUI after starting the game

### 3.1.3. Deployment

The application was deployed using Heroku [3]. Heroku is a cloud application platform that has its own command line tool. The Heroku command line tools includes support for deployment, logging, restarting the application and much more. Heroku provides an environment that allows different types of apps to be deployed with a couple basic commands. Our project is a Spring Boot application and thus supported by Heroku. With a GitHub repository already in place, deployment is as simple as pushing to the heroku master branch. Heroku then builds the application automatically and after building process is complete the application can be accessed at a determined URL (https://escapemaster.herokuapp.com/). For more information see the heroku reference: https://devcenter.heroku.com/categories/reference.

## 3.2. Communication with players

There are two communication channels implemented for the communication with players: SMS and email. For SMS, the telephone number of each team member is stored and also the messages to be sent. These messages are then coupled with an event to act as a trigger, so the messages are sent to players after a specific time or after starting a specific puzzle. Email messaging is the second communication channel which is implemented. There are two types of email communication. The first type is sending email messages after a specific amount of time that a puzzle is started or elapsed total game time (see events in 3.1.1). The second type is responding to messages that are sent by players. Players can ask questions by sending an email to a specific email address, the content is then parsed and sent to IBM Watson. Afterwards, Watson returns a response which contains an answer to the question asked. The answer is then put into a new message that will be sent back to the player.

The messages that are sent to the players occur when an event has been inputted into the system by the creator. For events there are two pieces of information required for the implementation of events, the last

solved puzzle of an individual player or team and the amount of time since that puzzle has been solved. The system is required to fire an event when both of these pieces of information are currently equal to the inputted event. And because the creator has the possibility to input an unlimited amount of events, the system must continuously check whether one of the events needs to be fired. Therefore, it was chosen to implement an observer pattern [13]. This design pattern can be seen in figure 3.4. Also it was decided to use a timer to check the events continuously.

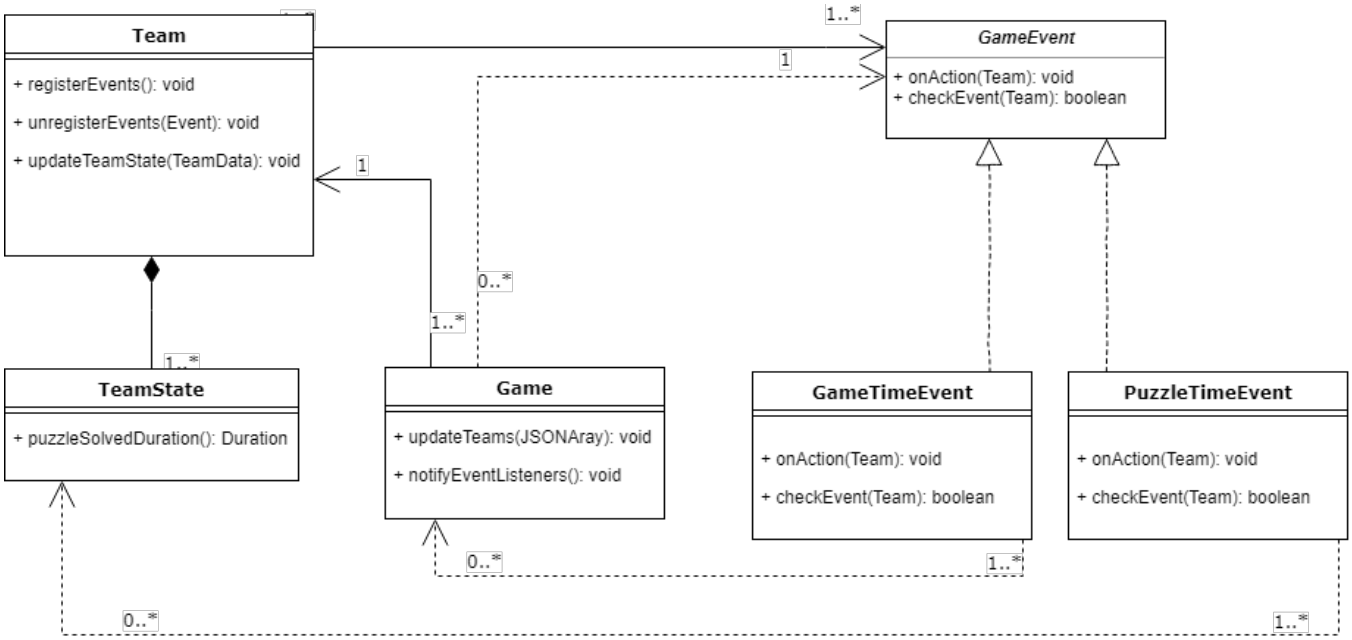



Figure 3.4: Observer design pattern

When a game is started the events are registered to all the teams. There are two different types of events that can be created, a GameTimeEvent that sends a message after a certain duration of the game and a PuzzleTimeEvent that sends a message after certain puzzles have been solved. An abstract class was used for this so that more events could later be implemented. The game also includes a timer that notifies events by checking if the event should be fired. This is done by looking into the game state of every team or checking the game time duration. Whenever an event has been fired and the team receives the message, the event is unregistered from the teams events that still need to occur. Whenever an event is not necessary anymore for instance, when an event should happen before a puzzle has been solved which functions as a hint, but the puzzle has already been solved, the event should still be fired and should not be unregistered. The reason for this is that the PuzzleTimeEvents are not only inputted to be hints, they can also be a message from 'the bad guys' saying for example, "Ha we have infected your computer!". And that kind of message should still be received by the players. Therefore, it was decided not to unregister events unless they have already been fired. Also, it is not likely that players unlock the next puzzle within the timeframe of receiving a hint. However, if the distinction should want to be made between a message and a hint after the project is completed, the implementation of unregistering events could be done in the following way. When a hint event is set to happen after $x$ amount of minutes when puzzle $i$ is solved, but the next puzzle, puzzle $i_{i+1}$ is also already solved, then the event is unregistered to prevent this event from firing. A diagram of the puzzles with events of the Social Hack escape room (the game that has been used to test the system) can be found in figure 3.5. To use this diagram as an example, after solving puzzle 0, an event is to occur after 3 min. But when puzzle 6 has already been solved before the 3 minutes have passed, the system should unregister the event to prevent it from happening. However, in the current implementation the system does not distinct between messages and hints and therefore, the system does not unregister events until occurrence.

Also a sequence diagram has been made to show how these events are checked and fired. This diagram can be seen in figure 3.6.

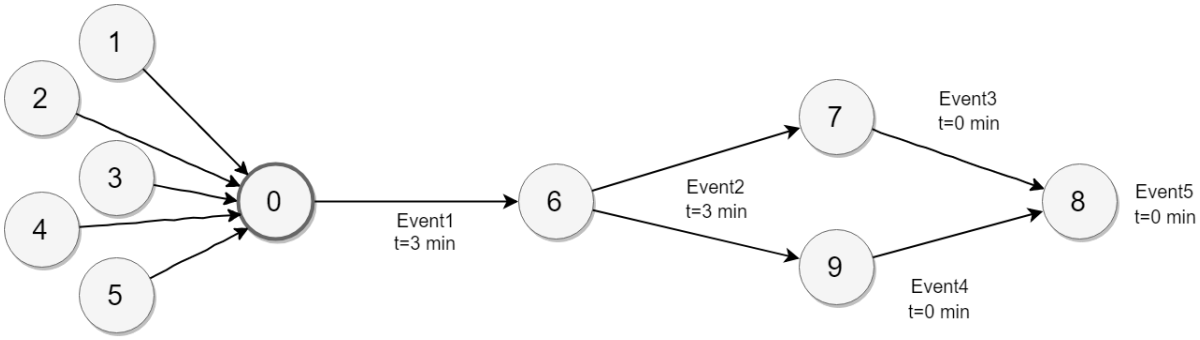



Figure 3.5: Puzzle diagram with events of the Social Hack escape room game. Example: Event1 must be fired after puzzle 0 has been solved for 3 minutes.

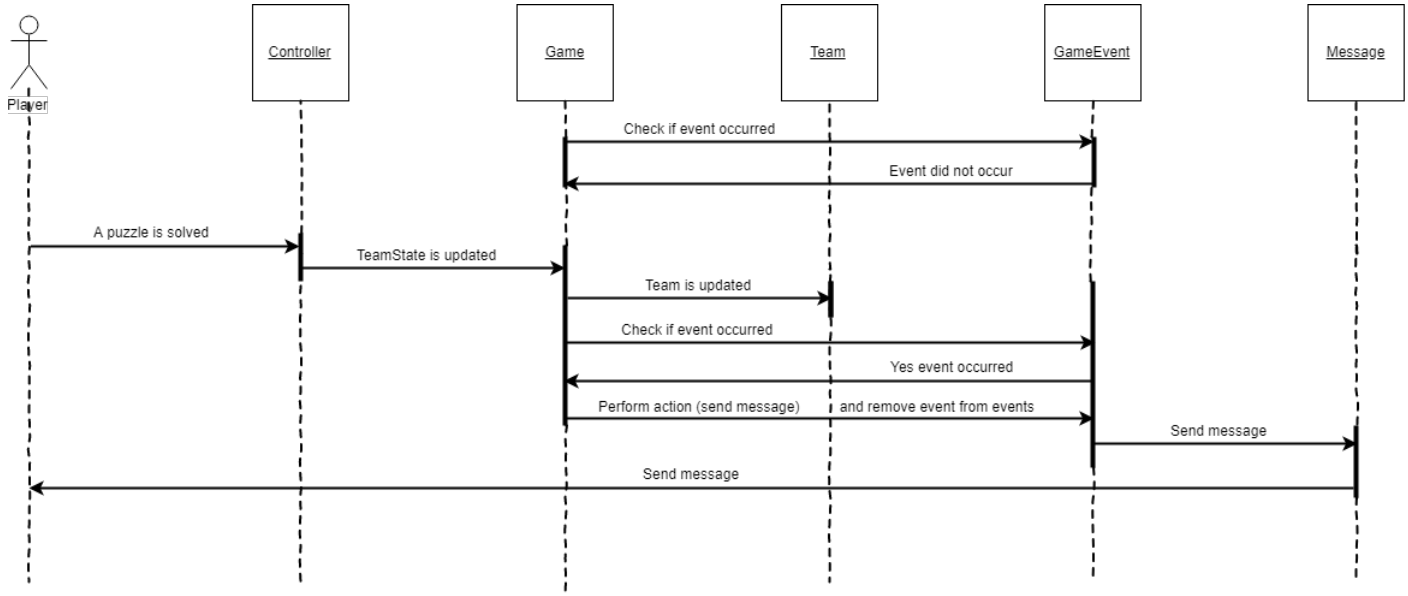



Figure 3.6: The sequence diagram of how the system handles events when a puzzle is solved by a team.

## 3.3. Automation of hints

Hints are coupled to the two designated event types, to reiterate: puzzle events and game events. Game events are triggered based on the elapsed game time. IBM Watson Assistant (Watson) is not used for this functionality, which does not need any language processing. Puzzle events have more logic. The Watson chatbot is created after the escape room information has been submitted. The creator is requested to enter example sentences and answers for each puzzle, called dialog bindings in the Escape-Master system. The example sentences are used to recognize the user's intent when sending an email. Knowledge of the exact internals of Watson is not necessary, intent recognition works better the more example sentences are provided by the user. It needs to be taken into account that there can be overlap between intents, and that example sentences need to represent the user's intent correctly.

A dialog tree is constructed based on these intents and the answers. A dialog tree is made up of 'nodes', that have specific conditions that trigger a context update and/or a reply. The structure of a dialog tree also influences the flow of the conversation. The answers are locked by what is called a 'folder', see figure 3.8. Watson requires a context variable to unlock this folder and reply with the answer that is pertinent when certain puzzles are solved. During runtime, when for example the Social Hack is being played, the context is updated with the information transmitted by Popup-Escape per team. This gives Watson the right context to give the correct answer that includes a hint.

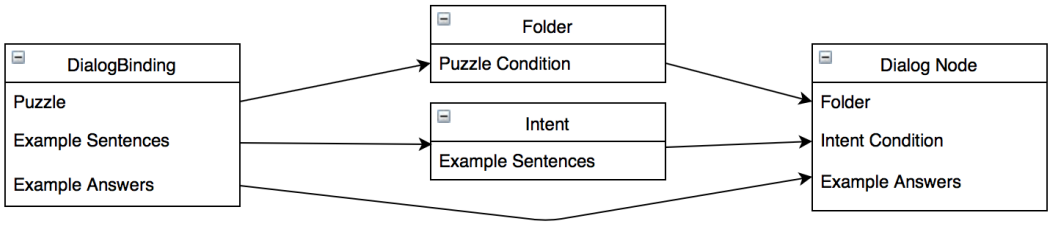



Figure 3.7: Diagram of the translation of a Dialog binding to the Watson context.

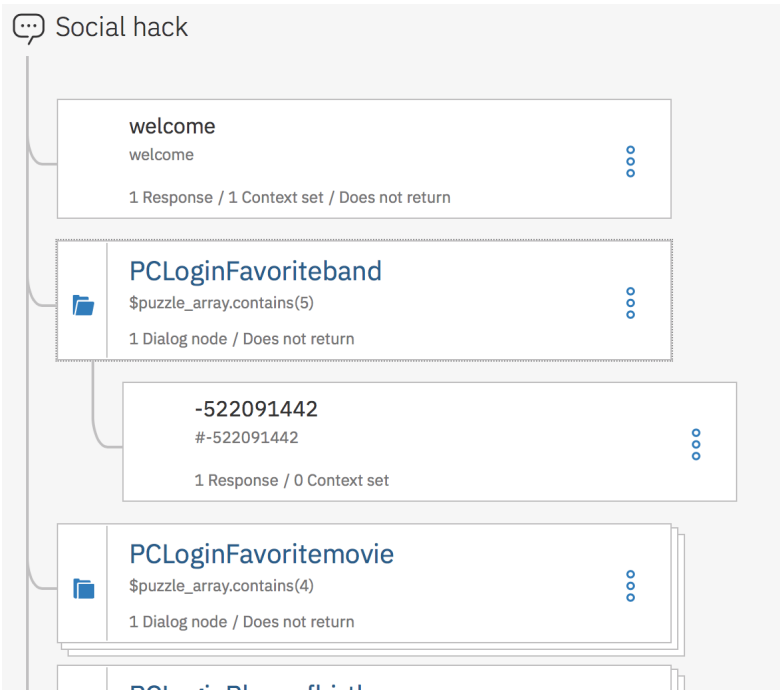



Figure 3.8: Part of the IBM Watson Assistant dialog tree with folders. The '$' indicates a context variable and the '#' an intent. The string of numbers is the hashcode of a dialog binding.

## 3.4. Persistence and modularization

The architecture of the escape room was at first viewed as being centered around puzzles. During the development process this view changed. It proved to be difficult to store the separate parts of an escape room in a database. The main benefit of having a modular architecture is that parts of an escape room can be recombined to create a new one, or an escape room can be updated or changed without effort.

A better technique was applied, namely the saving of the so called 'editor' of the escape room. The editor represents the state of the current input of the creator of the escape room through the user interface (UI), represented by the Spring HTML form. Loading an editor gives the state back to the UI and allows the creator to use the UI to manipulate the fields of the object. Since the editor is one object, it can also be stored as such. The main model still remained modular in an independent package. Thus the editor could be seen as a a collection of these modular parts. The consequence of this design change was that only one escape room can be played at a time.

# 4

# Testing

There were two approaches for testing the complete final product. First, system testing on the code level and secondly, testing towards quality assurance. Different methods were applied and are explained in full, along with the results for each type of method.

## 4.1. System testing

### 4.1.1. IBM Watson API

There are methods in the product that use the Watson API in order to create and manage the chatbot. These methods are within classes inside the chatbot folder. Tests were not made for these methods because it was not feasible nor reliable. Stubbing or mocking could be used, but its usefulness is very small in this case because most work is done by using the Watson API. The method itself is mostly about calling the Watson API code with the right parameters. The response from Watson could be tested but that would rely on the Watson server up time and the availability of an internet connection, making the test not reliable. For this reason, the decision was not to make tests for methods that use the Watson API. Instead, it was manually tested by using the web user interface on IBM Watson. Running the code locally and double checking if the expected results are seen on the website from Watson.

### 4.1.2. Creator system

The GUI of the creator system was mainly tested by using the GUI and see if the behaviour was as expected. Also a expert review was conducted which is discussed in 4.2.2. The data transmission over HTTP was tested using MockMvc.

### 4.1.3. Unit testing

The Model code base was tested using JUnit. Unit tests were applied to verify the logic for parsing and communication events. Testing the parsing logic involved creating test input and comparing this to test output. The parser should create the correct objects from the (JSON) formatted input and in the instance that the input was faulty throw an exception. Communication events were tested by updating the state of a test game and then checking if an event was triggered.

### 4.1.4. Results

For IBM Watson it was possible to verify that a method works by using the website, however this does not provide any evidence in terms of numbers to show for the client. On top of that, the Watson API relies on the service provided by IBM and if that service goes down, any tests that utilize that service will automatically fail.

For the creator system, testing the HTML data transmission prevented the introduction of several errors when the underlying data model was changed. The failing tests indicated that a change to the HTML or JavaScript code was needed to keep the GUI functioning. These changes were then implemented.

A coverage report was generated after running all tests. Certain packages were excluded from the test coverage report, as these were not tested as explained before. Some classes were also excluded, because they were simple Java beans without any testable logic. The non-trivial classes with important functionality do have an overall coverage percentage of 72%, see figure 4.1.

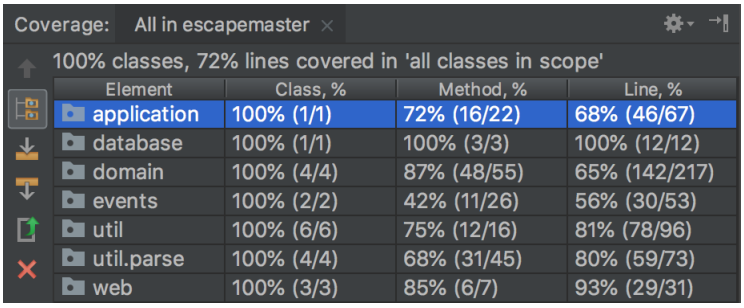



Figure 4.1: Test coverage report generated by IntelliJ IDEA [4].

## 4.2. Quality assurance

### 4.2.1. Code quality

The code was reviewed often to guarantee a high level of quality. The development process required scrutiny from other team members before code was accepted into the main code base of the Escape-Master project. When a new piece of code would be added, the continuous integration from Travis CI builds the project and runs all tests in an isolated environment. This feedback would be automatically relayed back to where the review of the code was executed. Using this setup our code should always pass all tests before it is accepted. The assurance of style quality was implemented using Checkstyle. This tool guaranteed that certain style requirements were met and otherwise gave a warning.

In collaboration with the TU Delft, the code was submitted to the Software Improvement Group (SIG), a private software assessment company. Their review concluded that the code was above average (3.5/5 stars). Code duplication and unit size were the main points of improvement. The number of tests could also be ameliorated. The complete recommendations can be found in the appendix D. After this first review the code base was updated accordingly and sent to SIG a second time to assure the improvements were implemented correctly.

### 4.2.2. Expert review

During this use test an expert was asked to use and review the product. Specifically, creating and inputting a new escape room is tested in this case.

**Participants** The participant of the expert review is solely the creator of the escape rooms. Which in this case is the client.

**Measures** The expert performs his regular tasks. After the review the expert provides feedback on the product. The feedback is analyzed by the development team and major problems are resolved.

**Procedure** During the review the expert inputted an entire escape room into the system such as, puzzles, events and characters. After the review, the expert provided extensive feedback. The expert had a chance to explain what problems had occurred and supplied feedback on the entire product.

### 4.2.3. Field studies

During the first field study 27 actual players have formed teams to play the full escape room, 'The Social Hack' which was earlier created by the game master. During the second field study 4 players have formed two teams to play 'The Social Hack' game.

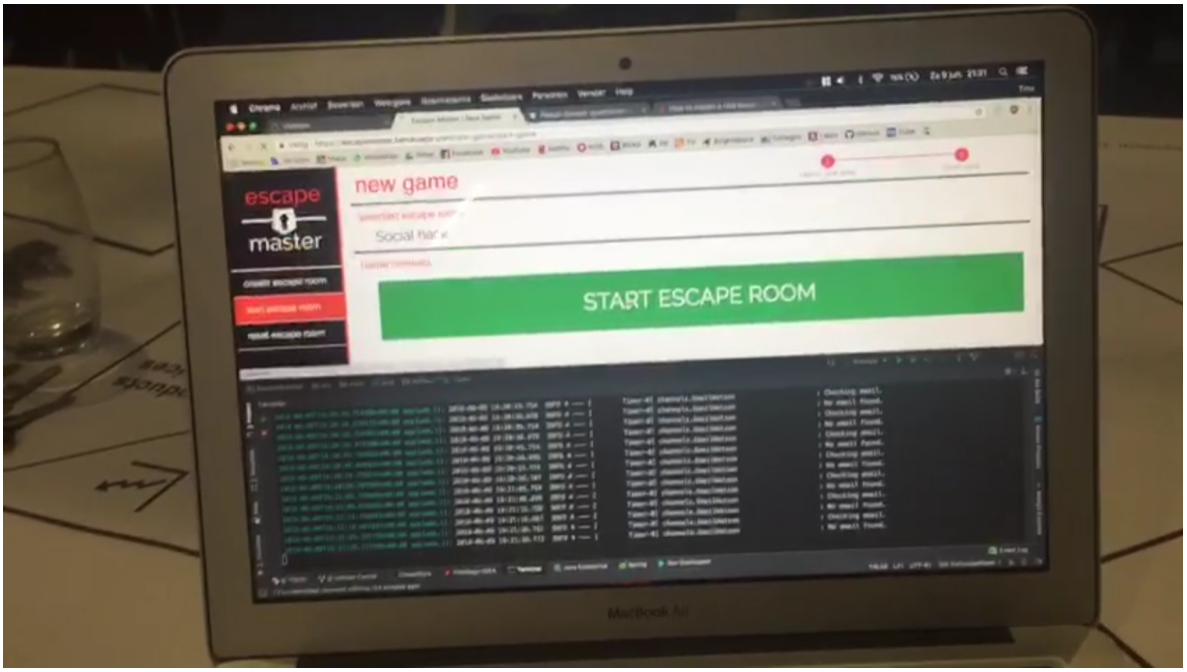



Figure 4.2: The first field study is about to start.

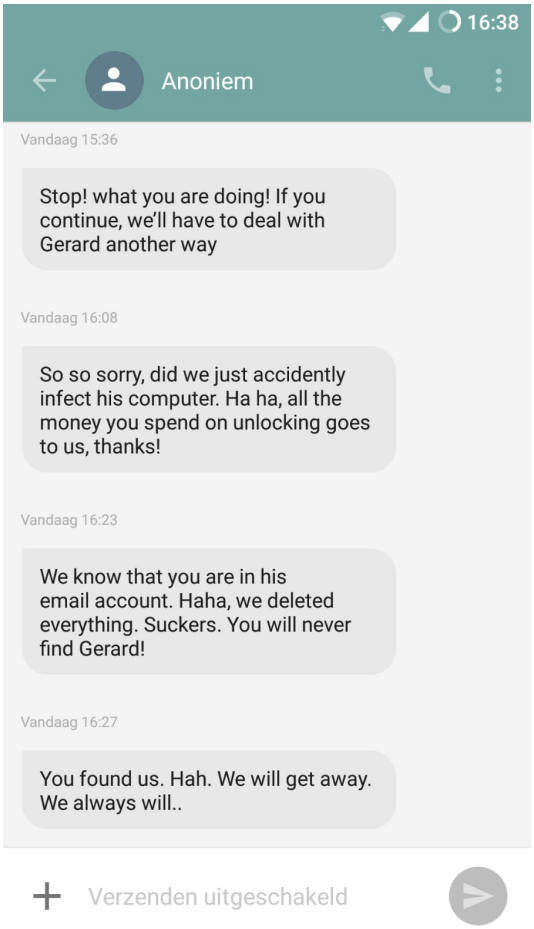



Figure 4.3: A players received SMS messages during the second field study.

**Participants** The participants of the field study were the players and the game master. This field study was also at the same time an expert review as the game master has a lot of experience. Both the game master and players need to evaluate the system to find problems with the usage of the product. The game master will make use of the already available system provided by Popup-Escape. The players brought their own telephones and each group has at least one laptop. During this field study there was a total of 27 participants in 5 teams and during the second field study a total of 4 participants in 2 teams.

Figure 4.4: The first field study and its participants.

Figure 4.5: A participant checking his email during the second field study.

**Measures** To analyze, evaluate and adjust the product where needed, all players were asked to fill in a questionnaire after the game. This questionnaire was made beforehand and created in a way that the biggest problems with the product will be found. The questionnaire can be found in appendix F. The collected data is presented in a graphical way, such as graphs to give a clear overview of the findings.

Another important measurement is the systems performance checklist. This list contains certain actions the system must complete in order to be a successful product. During the field studies the system was observed and it was validated that each item on this list was completed. These items can be found in table 4.1. The entire system is evaluated by analyzing the completed and uncompleted items on the checklist. Based on the completed items on the checklist the quality level of the end product can be demonstrated.

For the second field study data was collected for the measures shown in table 4.2 which were used for statistical analysis. The data described in the descriptions column was used to decide how well the system performed. To do this the recall was calculated by dividing the amount of correct responses by the amount of received responses of the players. And the precision was calculated by dividing the amount of correct responses by the amount of total responses.

**Procedure** All the participants met at the agreed upon location. The escape room was setup beforehand by the game master. The game master took care of all aspects of hosting the game and its players. Beforehand the players were asked to fill out a questionnaire as soon as the game was done.

**Goals** The most important goals of the field studies were the following:

- Check if the product requirements were met;
- Find bugs that were not discovered during unit testing;
- Test the system's stability under actual load by having multiple teams play the game;
- Gather data from players that might improve the response from the AI system;
- Gather feedback from the players on which the product can improve on;
- The client can get familiar with the product and see it in action.

### 4.2.4. Results

As previously stated, the quality assurance tool Checkstyle was used for our code base. At the end of the project no Checkstyle errors could be found in our code.

During the expert review, the client used the system to input an escape room and found that a few parts of the UI needed improving. The feedback received and the actions taken were the following:

- *When opening the editor, the user first had to click on the name field before they could start typing, which was not intuitive.*
  This was solved by adding autofocus to the appropriate fields;
- *It was not clear to the user how the step of adding links worked if the user had multiple puzzles that all needed to be solved before the players can advance to the next puzzle.*
  At the time the system did not support this specific case. The system was changed to accommodate this use case;
- *The user found it more intuitive to add the puzzle time events during the step of adding puzzles to the escape room.*
- *The user wanted to receive messages in the user interface to monitor the actions the system was taking.*
  A logger was added to the user interface that logs all actions that are relevant to the game master. The logs can be seen in figure 4.6.

Furthermore, it was found that some of the inner workings of the system were not clear from the start. This caused the user to define wrong behaviour in the system. Additional explanation was needed to prevent this from occurring again.

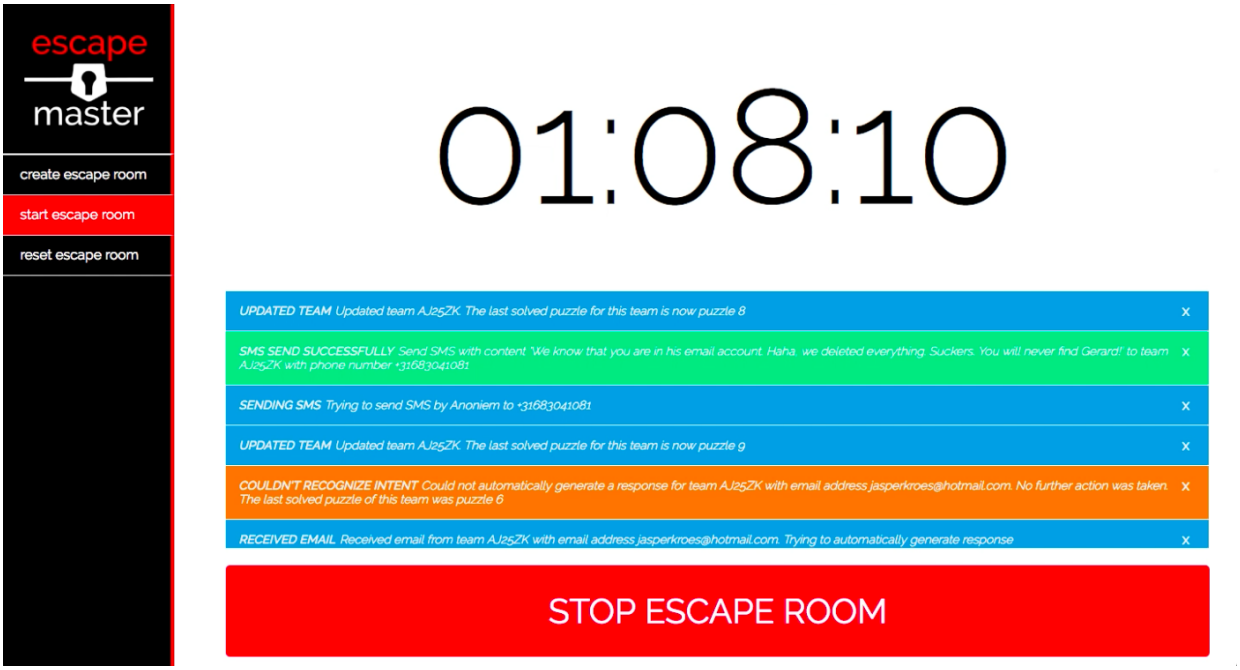



Figure 4.6: The logs were added after the review. These can be seen above the Stop Escape Room button.

As can be seen in the performance checklist in table 4.1, the first field study did not carry out to perform all the required actions. The human fallback system was not yet implemented, the emailing responses was not functioning and the SMS functionality was not working as intended. However, the second field study resulted in all required items completed in full, which indicates a fully functioning system based on the product requirements.

Table 4.1: The checklist.

| | **First field study** | **Second field study** | **Description** |
|---|---|---|---|
| Game Play | ~ | ✓ | Send players a predefined message after a certain amount of time has elapsed during a game (GameTimeEvent/ PuzzleTimeEvent) |
| | ✗ | ✓ | Respond to the players with a hint after receiving an email from the players |
| | ✓ | ✓ | Respond to the players with a message when receiving an undefined email from the players |
| Creator System | ✓ | ✓ | Input an escape room and save it to the database |
| | ✓ | ✓ | Input teams |
| | ✓ | ✓ | Start a game with the inputted teams and the selected escape room |
| | ✓ | ✓ | Update the state of each team during game play |
| | ✓ | ✓ | Display the logs of the game |
| | ✗ | ✓ | Human Fallback system |

Due to some bugs found during the first field study 57.9% of the participants did not receive any SMS messages. It was also shown that 80% of the participants sent an email to the fictional AIVD character. An interesting statistic was that of the interaction between players and the characters via SMS and email. The participants were asked if they were interacting with a human or a bot. The answers were spread evenly between bot, human, bot and human, and no clue (see figure 4.7). The second field study gave different results. The improvements made from the first study clearly yielded refinement. All participants now received SMS and email also performed much better, although sometimes human intervention was still necessary. The statistics of the second field study can be seen in table 4.2. And the full results of the both field studies and

the questionnaire can be found in appendix G.

In table 4.3, an example can be found of emails sent by players and corresponding responses of the system.

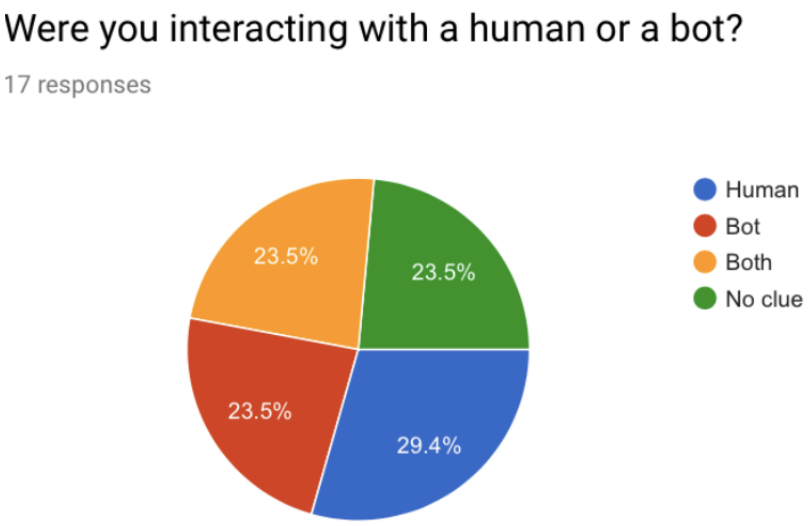



Figure 4.7: A result from the first field study.

Table 4.2: The measures used for the second field study and the results.

| **Descriptions** | **Measures** | **Statistics** | |
|---|---|---|---|
| | | **Team 1** | **Team 2** |
| By completion of the entire Social Hack game, a certain amount of SMS messages must be sent to each team | *5 SMS messages per team* | *5/5* | *3/3* |
| When a team sends an email for which the system has a response, the system must respond | *Recall = amount of correct responses / amount of received messages* | *Recall = 4/5 = 0.8* | *Recall = 3/3 = 1* |
| When the system responds to an email, it must give a correct response | *Precision = amount of correct responses / amount of total responses* | *Precision = 4/4 = 1* | *Precision = 3/3 = 1* |

Table 4.3: Examples of user sent emails and system generated responses during the second field study.

| **User input** | **Generated output** | **Correct output** |
|---|---|---|
| *"What is the place of birth of Gerard?"* | *"We think he is born in a small town south of Zutphen."* | Yes |
| *"What is the name of Gerards pet and where can we find it?"* | *"We called Faythe and she told us he has a nice picture of their pet in the background."* | Yes |
| *"msfhviratimnvoe iytaeotileaojib"* | (No answer generated.) | Yes |
| *"What is Gerards favourite movie? AJ25ZK"* | *"We really don't know, sorry. Maybe you should check the security codes."* | Yes |
| *"What does the safetycode for recovery questions mean?"* | *"Try shuffling all letters."* | Yes |

## 4.3. Discussion

The final product was tested sufficiently. Some components were not unit tested, but were tested manually and during an actual game with participants. The components that were unit tested had an overall test coverage of 72%. The product is of high quality because it kept a consistent coding style that helps with using good practices in code, since there were no Checkstyle errors in the master version of the software throughout the project. Besides, Travis CI also tested and built the software continuously, and only a passing build was on the Master branch.

The expert review was also helpful for the quality assurance of the product. After the expert review, the major problems were resolved and the client received additional explanation on how to input events as this had previously given some problems during the field study. This extra information was also placed in the UI where new events are inputted.

During the first field study 57.9% of the participants did not receive SMS messages. This was caused by a bug in the system. However, before the second field study, major improvements were made for SMS messages. SMS worked properly that time, and all players received SMS messages. During the first field study, email responses did not work properly due to IBM Watson being down for maintenance therefore, no data was collected. For the second field study, both SMS messaging and email messaging were improved. When the system replied, the responses were a 100% correct for both teams. When a team sent an email for which the system had a response, 80% of the messages were replied to for team 1, whilst for team 2, 100% of messages were replied to. 20% of the messages that were not replied upon, because the question was asked inside the subject field instead of the content field in the email. This was not implemented and could be improved for future field studies. The results from the question whether the participants were interacting with a human or a bot from the first field study is promising, because this shows that AI has the potential to replace the game master in a majority of settings however, human intervention is still required in other specific settings for instance, when no suitable response is available for a received message.

To conclude, the goals of the field studies have been fully achieved; The requirements have been checked and were all met, a few bugs were found and fixed, the systems stability was tested with ten teams, data was gathered to improve the response system, the system was improved based on feedback of the players, and the client has seen the product in action.

# 5

# Conclusion and discussion

## 5.1. Evaluation

**The first goal** of the project Escape-Master is to automate the role of the game master and create an engaging experience using a chatbot. The communication between players and the email and SMS system provide an adequate automation for the game master. Nevertheless, the game master must interfere in situations where the chatbot (IBM Watson Assistant) is not able to infer the intent of the player. During the two field studies the reactions of the participants surmised that the automated messages contributed to a rousing adventure. The true level of excitement is hard to measure and inherently subjective. Thus it was deemed more important that the client was satisfied with the use of the product. Consultation with the client concluded that the level engagement was improved by the employment of the product.

**The second goal**, is the modularization of escape rooms similar to Social Hack. The front-end interface and the accompanying back-end developed throughout the ten weeks that spanned this project gave the creator of escape rooms tools to make new ones by simple interaction with the system. The persistence of the escape rooms allows for loading and saving.

The product requirements that were defined in chapter 1.3.2 and whether they have been met and how, can be found in table 5.1. All Must and Should Haves were implemented, but none of the Could or Would Haves, as was the aim of the project. However, with more time available the Could Haves could be implemented as well. In the course of the project, prioritizing requirements was indispensable. The chief requirements were met, thus can be deduced that the project was a success.

Based on the results of the first and second field study it can be concluded that the system is well tested, as major issues were resolved before the second field study. This is shown in table 4.1, as all predefined functionalities are implemented and contained within the end product. Also, analyzing the statistics of the accumulated data during the second field study shows that the end product is a well functioning system that meets all the requirements.

## 5.2. Limitations

The product has certain limitations when using this in practice. The system worked well for English speakers but does not work yet for other languages. Although SMS works in the Netherlands, there was no research done whether it also works for foreign countries. The system works for digital puzzles, but currently there is no information tracked from non digital puzzles. IBM Watson was used as chatbot framework and relies on it for email replies, but the framework is not easily interchangeable unless the implementation is modified. The product works on the email server that Popup-Escape uses, however it is not easily adaptable for use for other email servers. Another limitation is about multi game hosting. It is possible to host multiple games by using multiple hosts, but it is limited to one game instance per host.

## 5.3. Ethical implications

The escape master system uses email addresses and telephone numbers of the players participating in an escape room game. This sensitive data is only used when the escape room is being played and provided by Popup-Escape. It could be used to send phishing or spam messages. The implications are negligible, because the same data was already collected and used by Popup-Escape. However, if the escape master system is not secure the data could be extracted. A malicious attacker would have to know when a game is started and have sufficient knowledge of the internals of the system to be successful. The effort for an attacker would be far greater than the value of the data. Therefore it would seem highly unlikely that someone would mount an attack. Moreover, the escape master system is hosted on Heroku, which guarantees security on every layer of their platform.

The stakeholders (creator, players) of the escape master system have disparate benefits. The players do not have a large increase in enjoyment of the game compared to how the escape room game was played without the escape master system. In the previous scenario creators would send emails and other message themselves. The creator has the chief advantage by using the escape master system. Now many more teams can play the same escape room and all receive messages relevant to the state of the game.

The natural language processing of Watson is trained on one billion Wikipedia words. The authors of these articles may not correctly represent the demographic of actual society. People with a more closely related background and thus language usage of these authors will get a better product. This negative effect can be negated by the creator. By inputting example sentence that have a mixed language usage, all players will get the same level of recognition.

## 5.4. Contributions and recommendations

The full experience of the client's escape room during the field study granted the client an acceptable attestation of what the Escape-Master system could contribute. The level of engagement of the game is assuredly increased by extra hints and messages sent via SMS and email. Also, currently the game master is responsible for responding to all emails received from the different teams, which can be a lot of work especially when the team sizes increase. Furthermore, another important aspect of the email messaging is the chatbot, as can be seen in the results of the questionnaire, the participants where not sure whether they were communicating with a bot or a human. This is an interesting result, which could be further explored and tested to find out what kind of experience this stipulates for the players.

Some features could still be implemented: the type and conditions of events can be extended for example, when a player visits a certain website a message could be sent.

Table 5.1: The completion of requirements and their description.

| | Requirements | Implemented | Description |
|---|---|---|---|
| Must Haves | Ability to support multiple teams at the same time | Yes | Multiple teams can play the escape room at the same time |
| | Detect when players need a hint, and communicate these hints using dialog by a fictional character through a text based medium, like chat or email | Yes | The players can ask for a hint by emailing a certain character of the game and the character responds with an email, however this is done at predefined moments of the game |
| | Possibility for the creator to input an escape room (order of the puzzles, hints, story, etc) into the system | Yes | The creator can input an entire escape room with multiple puzzles and events that can work as hints that is then saved in the database |
| | Possibility for the creator to create fictional characters with unique dialog | Yes | When inputting an escape room the creator also defines characters, with their channel (sms, email) and their dialogues |
| | Natural language interaction in English with fictional characters, by the chatbot | Yes | Yes when receiving an email by a player the chatbot responds to the player by checking in what game state that player is which unlocks a certain dialogue |
| | Retrieve and use game information (e.g. answers to puzzles, number of answer attempts) from existing Popup-Escape system | Yes | The game state of each team is updated whenever a change is made such as, the team has solved a puzzle |
| | Ability to reconstruct the current state of the game | Yes | With the information of the clients system all of the teams states can be saved to the system |
| | Ability to know which messages are sent by which team | Yes | When a message is received, the known email addresses are checked for each team, when the address is known the chatbot responds with the predefined dialogue, if the address is not known the system responds by asking for the teamcode |
| Should haves | Interaction with players via more than one medium (e.g. chat client, e-mail and social media) | Yes | SMS and email |
| | Ability to hold basic user information (e.g. email, telephone number) to link back the different forms of communication to the right team | Yes | The information is saved from all teams at the start of the game |
| | Human fallback system that makes it possible for the game master to answer questions that the AGM does not have an answer to | Yes | The responses are tracked and the can see in the logs what message has been responded to or not |
| | User friendly UI based on expert review | Yes | An expert review has been done and measures were take based on the received feedback |
| | Pro-active events based on the storyline and game progression | Yes | The GameTimeEvent sends a message when the duration of the game is at a certain point. The PuzzleTimeEvent sends a message when the players are playing a certain puzzle for a certain amount of time. The VisitedEvent sends a message when a player has arrived at a certain website |

# Bibliography


[1] Bot framework. URL https://dev.botframework.com/.

[2] Dialogflow. URL https://dialogflow.com/.

[3] Heroku: Cloud application platform. URL https://www.heroku.com/.

[4] Intellij idea. URL https://www.jetbrains.com/idea/.

[5] How to choose machine learning algorithms. URL https://docs.microsoft.com/azure/machine-learning/studio/algorithm-choice.

[6] Mongodb. URL https://www.mongodb.com/what-is-mongodb.

[7] What is interaction design?, . URL https://www.interaction-design.org/literature/topics/interaction-design.

[8] Design scenarios - communicating the small steps in the user experience, . URL https://www.interaction-design.org/literature/article/design-scenarios-communicating-the-small-steps-in-the-user-experience.

[9] Wit.ai, . URL https://wit.ai/.

[10] Getting started with wit.ai, . URL https://wit.ai/docs.

[11] Watson assistant the ai assistant for your business, Apr 2018. URL https://www.ibm.com/watson/ai-assistant/.

[12] Sameera A Abdul-Kader and John Woods. Survey on chatbot design techniques in speech conversation systems. *International Journal of Advanced Computer Science and Applications*, 6(7):72–80, 2015.

[13] Justin Albano. The observer pattern using java 8 - dzone java, Jan 2016. URL https://dzone.com/articles/the-observer-pattern-using-modern-java.

[14] Luka Bradeško and Dunja Mladenić. A survey of chatbot systems through a loebner prize competition. In *Proceedings of Slovenian Language Technologies Society Eighth Conference of Language Technologies*, pages 34–37, 2012.

[15] Rikke Dam and Teo Siang. Personas – a simple introduction. URL https://www.interaction-design.org/literature/article/personas-why-and-how-you-should-use-them.

[16] IBM. Does your business think?, Oct 2017. URL https://www.ibm.com/watson/.

[17] Michael Kriegel, M Lim, Ruth Aylett, Karin Leichtenstern, Lynne Hall, and Paola Rizzo. A case study in multimodal interaction design for autonomous game characters. In *3rd Workshop on Multimodal Output Generation*, pages 15–25. Citeseer, 2010.

[18] Henry Lieberman. Autonomous interface agents. In *Proceedings of the ACM SIGCHI Conference on Human factors in computing systems*, pages 67–74. ACM, 1997.

[19] Mindbowser. Chatbot market survey- 2017: Mindbowser info solutions, 2017. URL http://mindbowser.com/chatbot-market-survey-2017/.

[20] Brad A Myers and Mary Beth Rosson. Survey on user interface programming. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 195–202. ACM, 1992.

[21] Scott Nicholson. Peeking behind the locked door: A survey of escape room facilities. *White Paper available online at http://scottnicholson. com/pubs/erfacwhite. pdf*, 2015.

[22] Heather L O'Brien and Elaine G Toms. What is user engagement? a conceptual framework for defining user engagement with technology. *Journal of the Association for Information Science and Technology*, 59 (6):938–955, 2008.

[23] David Lorge Parnas. On the criteria to be used in decomposing systems into modules. *Communications of the ACM*, 15(12):1053–1058, 1972.

[24] Silvia Quarteroni. Natural language processing for industry. *Informatik-Spektrum*, 41(2):105–112, Apr 2018. ISSN 1432-122X. doi: 10.1007/s00287-018-1094-1. URL `https://doi.org/10.1007/s00287-018-1094-1`.

[25] Alan M Turing. Computing machinery and intelligence. In *Parsing the Turing Test*, pages 23–65. Springer, 2009.

# Appendices

# A

# Project description

Popup-escape is a young company/startup started by a Delft Computer Science student two years ago. It started out as a hobby and it is now grown to a full-time job with multiple part-time employees. (Also some Computer Science students). We have designed and made about 30+ escape rooms over the past two years and at least 3000 people have played the different escape rooms.

Popup-Escape has created a social hack game, a game where the players have to hack a computer using social engineering. They have to explore and find information on the internet using social media, (fake) websites, calling phone numbers and more. This is combined with Physical puzzles and tasks the player have to complete. The players play in groups of 3-5 against each other, and have a live scoreboard of the game up whilst playing. The software system behind this game is pretty complex, following users across different websites and systems, checking their actions (online and offline) whilst updating the live scoreboard. Whilst the game is a lot of fun using real world actions in a game, we would like to add another dimension: Interacting with AI characters to enhance the social engineering experience. That's where you come in. We would like you to research the possibilities for such an AI system which feels natural and real. Furthermore, we would like you to implement such a system with an API so that we(or you, if you'd like) can integrate it with our current working system. Popup-Escape offers:

- A lot of fun: You are going to play multiple escape rooms during the project for inspiration and research
- A informal (student-like) work environment and great lunch
- Flexibility: it is not required to work at the office all the time and the working hours are determined by you
- Working with technologies you like and think are the best
- The possibility to work on games which are going to be played by real users. You can see your software in action!

# B

# Info sheet

# Info sheet

**Title of the project:** Real life social hacking game
**Name of the client organization:** Popup-Escape
**Date of the final presentation:** 4 July 2018
**Description:** The client of this project was Popup-Escape; a company that creates and hosts personalized escape rooms. The client wanted a product that automates the role of a game master of an escape room. The game master is responsible for supplying hints and parts of the narrative. Communication of the automated host pertains two types of media: email and SMS. The product was designed to represent a modularization of escape rooms. Consequently, similar escape rooms can be modeled and automated by the product. During the research phase, the product was found to be unique. Therefore the team researched which components of the system were needed to create a fully functional product. The Scrum methodology was employed for the development of the product. An unexpected challenge of this project was the integration of the major components in order to create an usable escape room model for the client and an engaging experience for the players. The quality of the product was guaranteed by unit testing the product and field testing the escape room game with a total of twenty-seven customers of the client. A possible improvement of the product would be a more elaborate model of escape rooms. This would include extending the possible media with which the product communicates with the players and the conditions on which messages are sent. Popup-Escape will use this product when playing future escape rooms comparable to the Social Hack.

**Members of the project team:**
All members contributed to the final report and the final presentation.


Name: Timo van Asten
Interests: Psychology, programming
Role & contribution: Communication, Front-end developer

Name: Bernard Bot
Interests: Ethics, logic
Role & contribution: Lead programmer, Back-end developer

Name: Björn Ho
Interests: Programming, algorithms
Role & contribution: Editor, Back-end developer

Name: Jael Lopez Kuchlin
Interests: Biomedical engineering, reasoning and logic, problem solving
Role & contribution: Management and organization, Back-end developer

**Client**
Name: Jan-Willen Manenschijn
Affiliation: Owner Popup-Escape

**Coach**
Name: W.P. Brinkman
Affiliation: Interactive Intelligence Group TU Delft

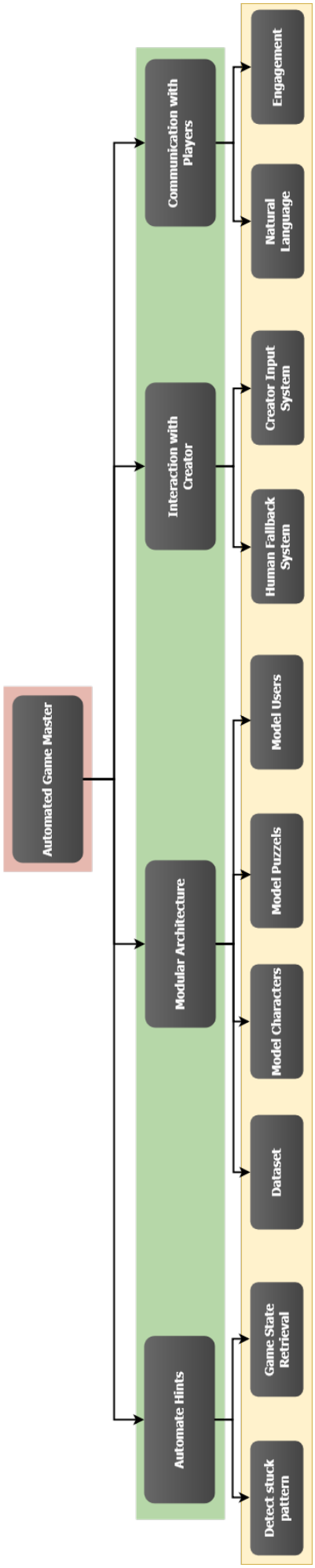
**Contact**

| | |
|---|---|
| Timo van Asten | timovanasten@gmail.com |
| Bernard Bot | bernardo.reinier@gmail.com |
| Björn Ho | bjornho@outlook.com |
| Jael Lopez Kuchlin | lopez.jael@gmail.com |


The final report for this project can be found at: `https://repository.tudelft.nl`

# C

# Roadmap


Automated Game Master
Automate Hints
Detect stuck pattern
Game State Retrieval
Modular Architecture
Dataset
Model Characters
Model Puzzels
Model Users
Interaction with Creator
Human Fallback System
Creator Input System
Communication with Players
Natural Language
Engagement

# D

# SIG Recommendations

## D.1. First feedback

De code van het systeem scoort 3.5 sterren op ons onderhoudbaarheidsmodel, wat betekent dat de code marktgemiddeld onderhoudbaar is. We zien Duplication en Unit Size vanwege de lagere deelscores als mogelijke verbeterpunten.

Voor Duplicatie wordt er gekeken naar het percentage van de code welke redundant is, oftewel de code die meerdere keren in het systeem voorkomt en in principe verwijderd zou kunnen worden. Vanuit het oogpunt van onderhoudbaarheid is het wenselijk om een laag percentage redundantie te hebben omdat aanpassingen aan deze stukken code doorgaans op meerdere plaatsen moet gebeuren.

In jullie project zit er bijvoorbeeld duplicatie tussen de verschillende event-classes, zoals PuzzleTimeEvent, VisitedEvent, GameTimeEvent, en dergelijke. Die classes bestaan vooral uit data en minder uit gedrag, maar desondanks zou je de hoeveelheid "boilerplate" code zoveel mogelijk willen beperken. Je zou er bijvoorbeeld voor kunnen kiezen om alle parameters in eerste instantie op een defaultwaarde te zetten (door in de interface GameEvent een default method te maken).

Voor Unit Size wordt er gekeken naar het percentage code dat bovengemiddeld lang is. Het opsplitsen van dit soort methodes in kleinere stukken zorgt ervoor dat elk onderdeel makkelijker te begrijpen, te testen en daardoor eenvoudiger te onderhouden wordt. Binnen de langere methodes in dit systeem, zoals bijvoorbeeld, zijn aparte stukken functionaliteit te vinden welke ge-refactored kunnen worden naar aparte methodes.

Een voorbeeld uit jullie systeem is GraphFactory.createGraph(). Deze methode is nog niet zo lang, maar er is wel een risico dat de methode erg groot gaat worden op het moment dat de graph in de toekomst ingewikkelder wordt. Je wilt de verschillende delen van het opbouwen van de grapgh eigenlijk naar aparte methodes uitsplitsen.

De aanwezigheid van testcode is in ieder geval veelbelovend. De hoeveelheid tests blijft nog wel wat achter bij de hoeveelheid productiecode, hopelijk lukt het nog om dat tijdens het vervolg van het project te laten stijgen.

Over het algemeen is er dus nog wat verbetering mogelijk, hopelijk lukt het om dit tijdens de rest van de ontwikkelfase te realiseren.

## D.2. Incorporation of feedback

The feedback from feedback noted two major points: unit size and duplication. In the initial release of our project, that was uploaded to SIG, we still had a lot of code that was not really used or only partially. This code was cleaned up: parts were removed and other parts were given full functionality. The amount of tests also increased accompanying this development. The largest change was the separation of the parsing logic and the actual escape room model. This resulted in multiple packages with disparate testing and logic. The project as a whole became more maintainable and easier to understand.

## D.3. Second feedback

In de tweede upload zien we dat het project ongeveer even groot is gebleven. De score voor onderhoudbaarheid is in vergelijking met de eerste upload gestegen.

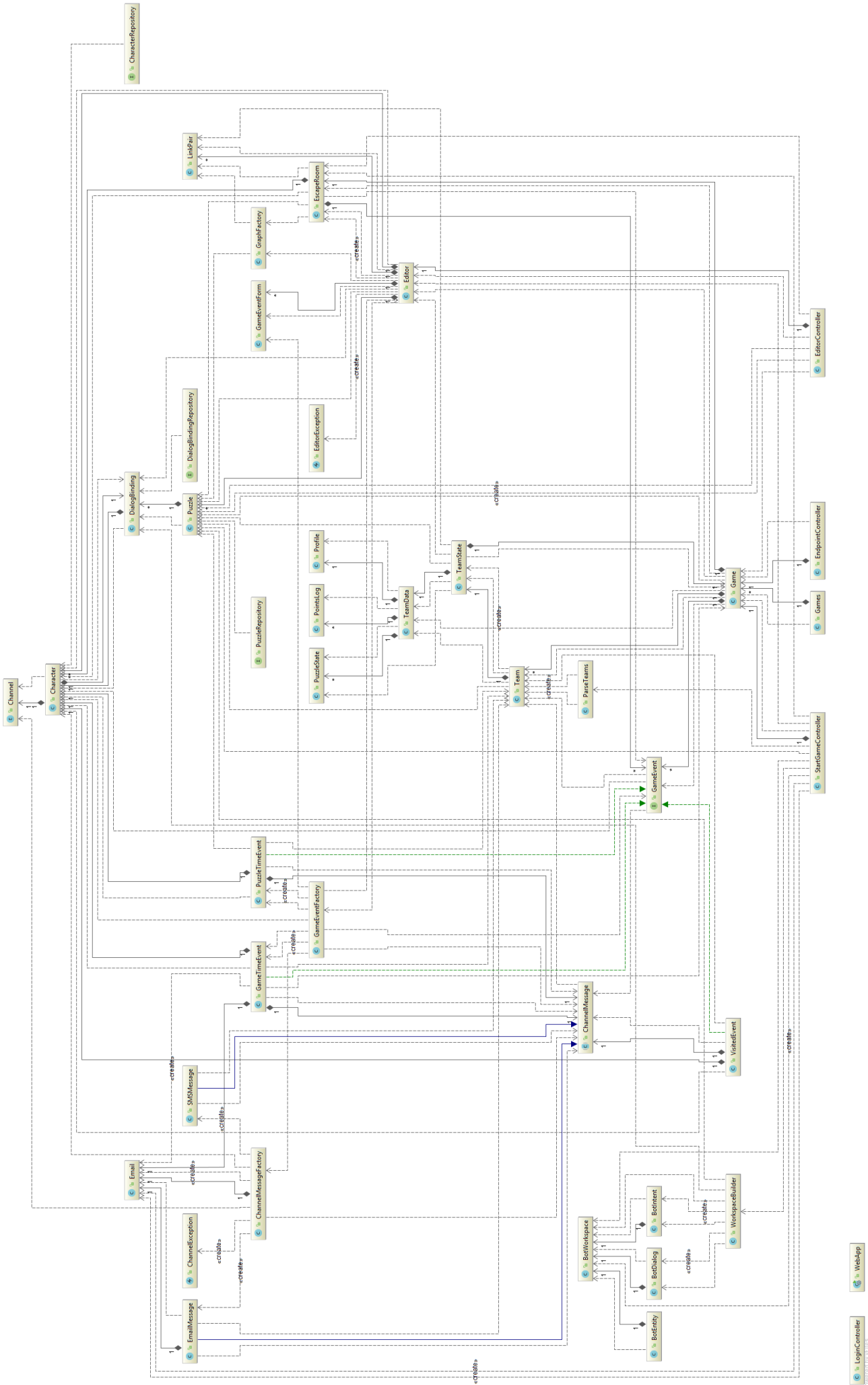
We zien bij Duplication een grote verbetering, niet alleen in de genoemde voorbeelden, maar ook elders in de code. Hulde! Bij Unit Size zien we ook een verbetering, maar iets minder groot en structureel dan bij Duplication.

Naast de toename in de hoeveelheid productiecode is het goed om te zien dat jullie ook nieuwe testcode hebben toegevoegd. De hoeveelheid tests blijft nog wel wat achter bij wat je gezien het volume van de productiecode zou verwachten, helaas is het niet helemaal gelukt om de achterstand in te lopen.

Uit deze observaties kunnen we concluderen dat de aanbevelingen uit de feedback op de eerste upload zijn meegenomen tijdens het ontwikkeltraject.

# E

# UML


CharacterRepository
LinkPar
EscapeRoom
GraphFactory
GameEventForm
Editor
EditorController
EditorException
DialogBindingRepository
DialogBinding
Puzzle
Profile
PointsLog
PuzzleState
PuzzleRepository
TeamData
TeamState
Team
ParseTeams
Game
EndpointController
Games
StartGameController
Channel
Character
GameEvent
PuzzleTimeEvent
GameEventFactory
GameTimeEvent
ChannelMessage
VisitedEvent
SMSMessage
Email
ChannelMessageFactory
ChannelException
EmailMessage
BotWorkspace
BotIntent
BotDialog
BotEntity
WorkspaceBuilder
WebApp
LoginController
«create»

# F

# Questionnaire

# Popup-Escape questionnaire

Hello there! We hoped you enjoyed the escape room. While playing the escape room, we tested an experimental system that sends you SMS messages when you solve puzzles and answers the questions you send via mail automatically. We are curious if you experienced any problems with the system so we can improve it. We would also love to hear your thoughts or feedback!

* Required

1. **Did you or your team members receive any SMS messages?** *
   *Mark only one oval.*

   - ◯ Yes
   - ◯ No

2. **Did you or your team members email any questions to the AIVD?** *
   *Mark only one oval.*

   - ◯ Yes
   - ◯ No

3. **Did you encounter any of these problems?** *
   *Check all that apply.*

   - ☐ The responses to the emails I sent where not helpful
   - ☐ I did not receive a response after I sent an email
   - ☐ I did not receive a hint when I asked for one
   - ☐ I received a hint while I did not ask for one
   - ☐ I received messages about puzzles that I was not yet trying to solve
   - ☐ I received messages about puzzles that I already solved
   - ☐ I did not encounter any of these problems problems
   - ☐ Other: _____

4. **Could you elaborate on the problems you encountered?**

   _____

   _____

   _____

   _____

   _____

5. **If you interacted with the AIVD via email during the game, how would you rate your experience interacting with the AIVD?**
*Mark only one oval.*

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| The interaction was not at all useful | ◯ | ◯ | ◯ | ◯ | ◯ | ◯ | ◯ | ◯ | ◯ | ◯ | The interaction was useful |

6. **Do you have any suggestions for improvement for us?**

_____

_____

_____

_____

_____

7. **Were you interacting with a human or a bot?**
*Mark only one oval.*

- ◯ Human
- ◯ Bot
- ◯ Both
- ◯ No clue

8. **What was your teamcode?**
(If you have this available)

_____

# G

# Questionnaire results

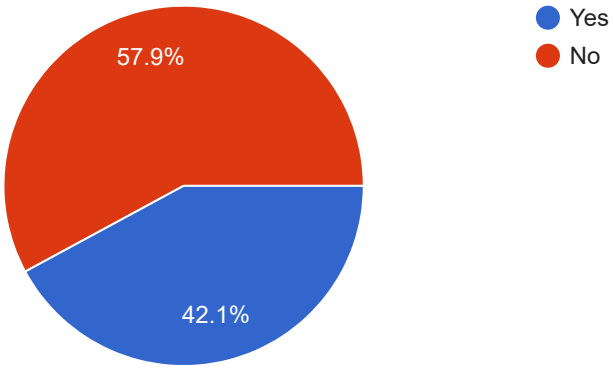
## G.1. Field study 1

# Popup-Escape questionnaire

19 responses

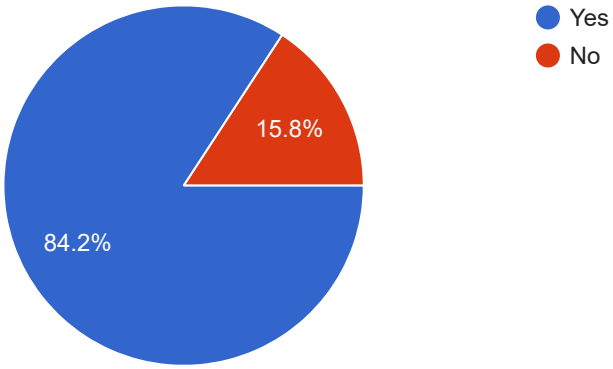
## Did you or your team members receive any SMS messages?

19 responses



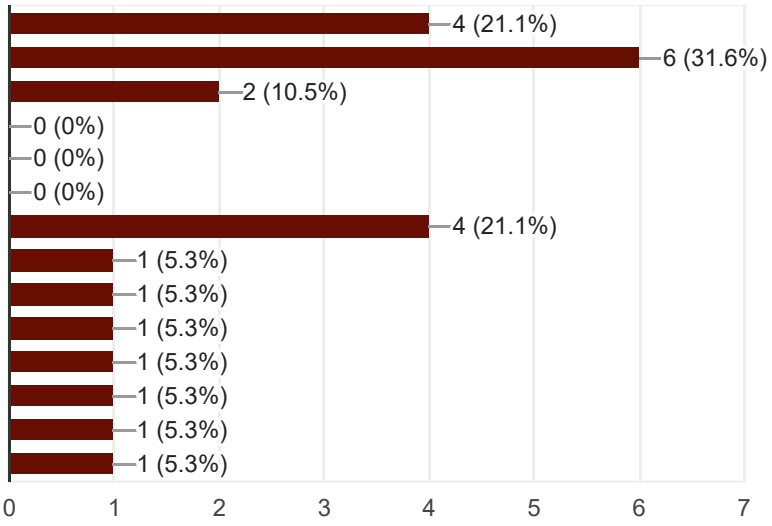

## Did you or your team members email any questions to the AIVD?

19 responses




## Did you encounter any of these problems?

19 responses



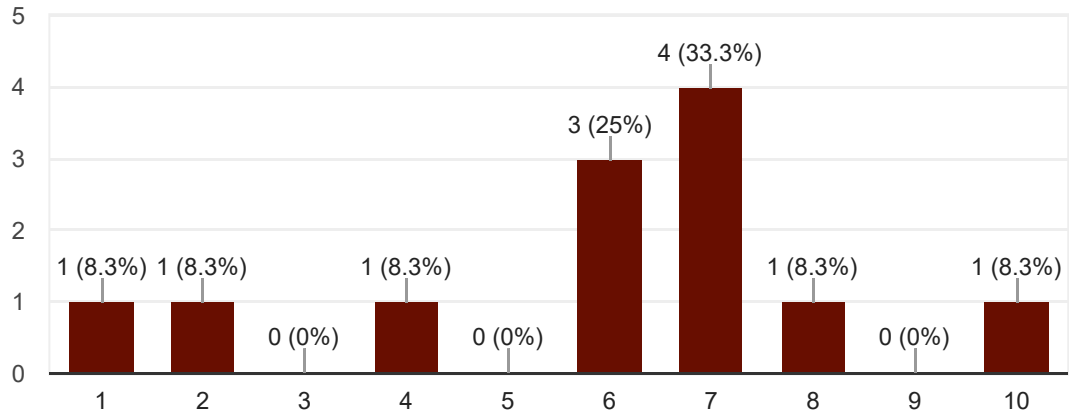

## Could you elaborate on the problems you encountered?

9 responses

| |
|---|
| We send an e-mail to the AIVD but did not recieve a response |
| Suggestion that we needed to include the team reference, after including it no more response.. maybe that was the plan? |
| Problem on BTC wallet |
| The content of the message we never reveived came too late |
| Foutmelding in mail. Deed het niet |
| See above |
| The code die the Bitcoin did not work |
| Didn't work when clicking submit on smartphone |
| SMS service did not work. |

## If you interacted with the AIVD via email during the game, how would you rate your experience interacting with the AIVD?

12 responses



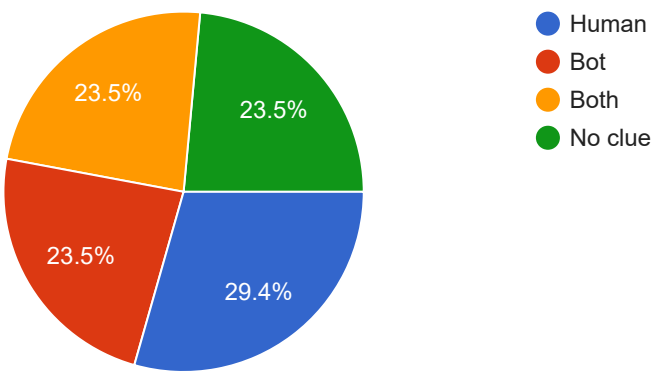

## Do you have any suggestions for improvement for us?

5 responses

| |
|---|
| Make sure everyone gets the text at the same moment! |
| Get the sms system working plz |
| Te lastig voor anderen? |
| Put some evidences in the answers of the e-mail |
| Didn't work when clicking submit on phone |

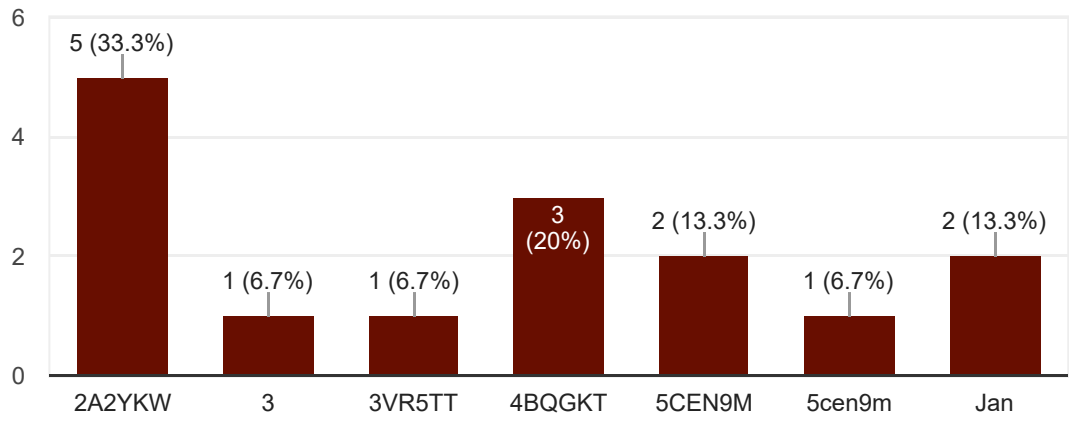
## Were you interacting with a human or a bot?

17 responses



## What was your teamcode?

15 responses



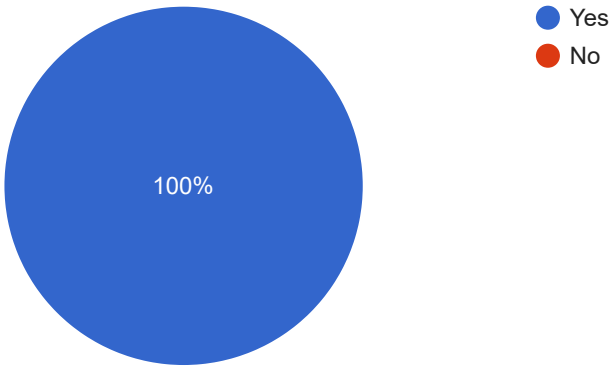
## G.2. Field study 2

# Popup-Escape questionnaire

4 responses

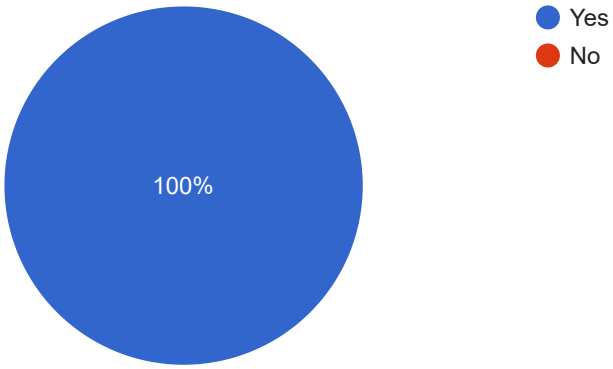
## Did you or your team members receive any SMS messages?

4 responses



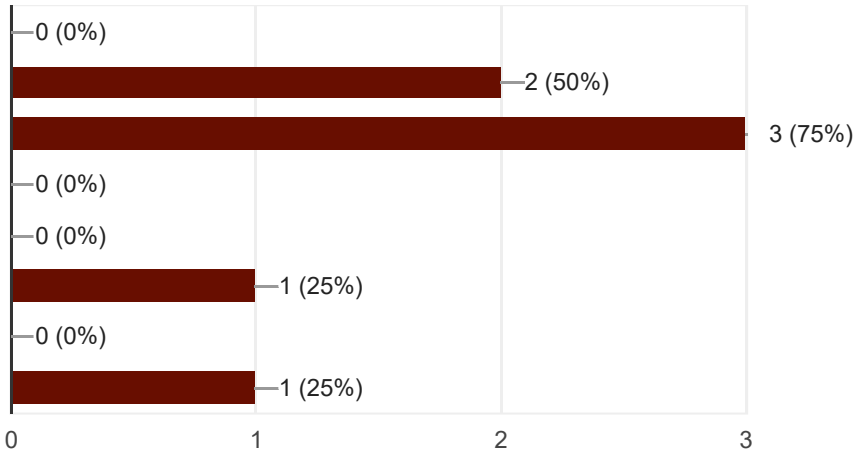

## Did you or your team members email any questions to the AIVD?

4 responses




## Did you encounter any of these problems?

4 responses




## Could you elaborate on the problems you encountered?

3 responses

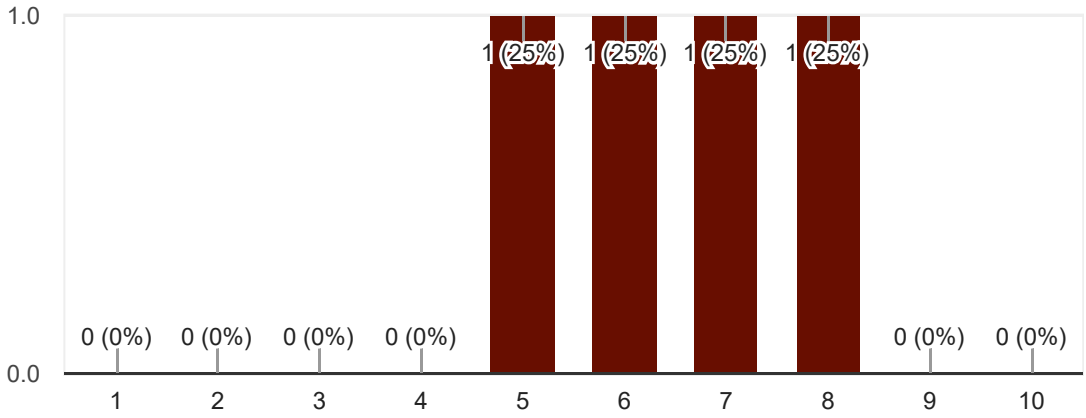
We asked a lot but did not get a lot ):

I dont think receiving the same hint multiple times is very useful.

I only received a response on 1 mail, the other mails I sent were not answered

## If you interacted with the AIVD via email during the game, how would you rate your experience interacting with the AIVD?

4 responses



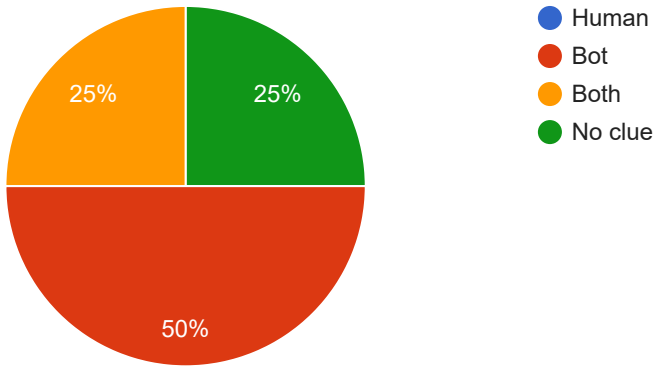
## Do you have any suggestions for improvement for us?

2 responses

| |
|---|
| I did not receive a hint for the final part of the escape room, even though I sent an email about it |
| Better mail responses |

## Were you interacting with a human or a bot?

4 responses




## What was your teamcode?

4 responses

| |
|---|
| PKL53Z |
| AJ25ZK |
| PKL nog iets |
| Jasper en Lisette |

# H

# Project Plan

# Escape-Master

## Project Plan

by

**T. van Asten**
**B.R.A. Bot**
**B.I.Y.L. Ho**
**J.R.R. Lopez Kuchlin**

| **Name** | **Email** |
|---|---|
| Timo van Asten | t. vanasten@student.tudelft.nl |
| Bernard Bot | b.r.a. bot@student.tudelft.nl |
| Björn Ho | b.i.y.l. ho@student.tudelft.nl |
| Jael lopez Kuchlin | j.r.r. lopezkuchlin@student.tudelft.nl |


| | |
|---|---|
| Project duration: | April 23, 2018 – July 6, 2018 |
| TU Coach: | W.P. Brinkman |
| Client Advisor: | J.W. Manenschijn |
| Bachelor Project Coordinator: | O. Visser |

TU Delft

# Contents


**1 Introduction** **1**

**2 Project Scope** **2**

2.1 Project description . . . . . 2
2.2 Components . . . . . 2
2.3 Evaluation . . . . . 3

**3 Project management** **5**

3.1 Numbers . . . . . 5
3.2 TU Coach . . . . . 5
3.3 Client advisor . . . . . 5
3.4 Team . . . . . 6
3.5 Communications . . . . . 6
3.6 Project team directory . . . . . 7

**4 Project Timeline** **8**

4.1 Deliverables. . . . . 8
4.2 Schedule . . . . . 8
4.2.1 Tasks . . . . . 10

# 1 Introduction

This Project Plan will provide a definition of the project, including the project's goals and objectives. Additionally, the Plan will serve as an agreement between the following parties: TU Coach, Client Adviser, Project Team and BEP coordinator.

**The Project Plan defines the following:**

- Project scope
- Project management
- Roles and responsibilities
- Project timeline

**Background information** Escape rooms are an emerging market. Solving puzzles under time pressure with friends or co-workers is fun to do. Popup-Escape provides this service primarily in Delft and around the Netherlands. They currently have two main products and are developing more in the future.

One of these two products is the 'Social Hack', a simple escape room where teams compete to find the location of a fictional character. The answer of one puzzle leads to another puzzle. Players become immersed in the story by finding out more about the person they are tracking down.

The puzzles of the 'Social Hack' are mostly text based, many are presented in a online, digital environment. This leads to a natural link to automation. Currently a human supervisor tries to guide the escape room challenge into the right direction. This task may be automated by using an AI coupled with a chat interface to help people.

The AI/chatbot will be designed with the intent of reuse. The main components of an escape room will be identified and represented in a object oriented manner. The objective will be to make the creation and design of escape rooms be analogous to the creation of a chatbot corresponding to that escape room.

# 2

# Project Scope

This chapter defines the project description and deadlines. Also the evaluation per subject are shown here.

## 2.1. Project description

Popup-escape is a young company/startup started by a Delft Computer Science student two years ago. It started out as a hobby and it is now grown to a full-time job with multiple part-time employees. (Also some Computer Science students). We have designed and made about 30+ escape rooms over the past two years and at least 3000 people have played the different escape rooms.

Popup-Escape has created a social hack game, a game where the players have to hack a computer using social engineering. They have to explore and find information on the internet using social media, (fake) websites, calling phone numbers and more. This is combined with Physical puzzles and tasks the player have to complete. The players play in groups of 3-5 against each other, and have a live scoreboard of the game up whilst playing. The software system behind this game is pretty complex, following users across different websites and systems, checking their actions (online and offline) whilst updating the live scoreboard. Whilst the game is a lot of fun using real world actions in a game, we would like to add another dimension: Interacting with AI characters to enhance the social engineering experience. That's where you come in. We would like you to research the possibilities for such an AI system which feels natural and real. Furthermore, we would like you to implement such a system with an API so that we(or you, if you'd like) can integrate it with our current working system. Popup-Escape offers:

- A lot of fun: You are going to play multiple escape rooms during the project for inspiration and research
- A informal (student-like) work environment and great lunch
- Flexibility: it is not required to work at the office all the time and the working hours are determined by you
- Working with technologies you like and think are the best
- The possibility to work on games which are going to be played by real users. You can see your software in action!

## 2.2. Components

**The main components of the product:**

- The product is delivered on 22 june 2018
- The product models a generic escape room
- The product can create a game host for an escape room
- The product is implemented using the Scrum methodology

- The product is tested with a minimum code coverage of 70%
- The code of the product is maintained with Github
- The product makes use of the SIG input to improve the quality
- The client can use this product with existing escape rooms and also future escape rooms
- The product can reply with messages when given questions by the player
- The product can give hints when a player is having problems with a puzzle

**Tools that are used during the product:**

- ShareLaTeX
- Google Drive
- Git
- GitHub
- Travis CI (preliminary, maybe another tool will be used)

## 2.3. Evaluation

- Research
  - Were the references appropriate and complete?
  - Did the research cover the possible alternatives for major decisions about approach, design, and implementation?
  - Were the project decisions grounded in a comparative analysis of the alternative solutions?
- Process
  - Did the process run smoothly and on schedule? Did the communication between team members support quick resolution of unexpected challenges?
  - Did the team make good use of software development methodology?
  - Did the team work independently? Was the team pro-active in seeking help when needed?
- Product
  - Did the system fulfill the original system specifications? Was the product well tested?
  - Was the code of high quality? Did the team take advantage of the SIG input?
  - Will/can the Client use the product (has the Client's goal been achieved)? Does it advance the state of the art or bring a new application to market?
- Report
  - Was the report well structured and clearly written?
  - Did the report present a complete picture of the project (i.e., no important phases in the project trajectory where omitted)?
  - Did the report provide motivation that the project was successful?
- Presentation
  - Was the presentation well structured and clear (including demo)?
  - Did the presentation convince the audience that the project was substantial, well motivated, and successful?
  - Did the team demonstrate their mastery of the material during the Q&A?

The evaluation in each of the above five dimensions should take the difficulty into account, e.g:

- Do existing (e.g., off-the-shelf) solutions to the problem already exist?
- Does the project have multiple facets? Were many decisions required?
- Did the project require that the team make use of a diverse skill set?

# 3

# Project management

This chapter explains the entire project management and all important numbers of the project. The team roles are defined and also the client and coach. Also, how the communication between team members and the client and coach will be managed is shown.

## 3.1. Numbers

**Ten important numbers:**

1. 3-4 students per team
2. 2 advisers (1 client adviser and 1 TU coach from EEMCS)
3. 1 project plan agreed upon between TU coach and team (“plan van aanpak”)
4. 2-week research phase with 10-page research report (from research phase)4
5. 2 submissions of code to the Software Improvement Group (SIG) for evaluation
6. 10-11 total weeks of work (420 hours)
7. 30-50 page final report (delivered to the committee 7 days before the presentation)
8. 1 A4 Info sheet that summarizes the project highlights (included as an appendix in the final report)
9. 3 Bachelor Project Committee members (1 TU Coach, 1 client, and 1 Bachelor Project Coordinator)
10. 30-minute final presentation (including demo)

## 3.2. TU Coach

The TU coach for this project is **Willem-Paul Brinkman**. His role is to represent the educational interests of the TU Delft. He acts as a guardian of the learning objectives of the bachelor project. The primary function of this role is guiding the bachelor students in applying their skills and knowledge within the bachelor project. The TU Coach should support the team in choosing and making use of the appropriate software development methodology. Practically, the TU Coach should offer support in keeping the bachelor project running along its timeline and converging to a timely, finished project. Typical activities of the Coach are: Approving the project plan, meeting regularly with the team, and giving any needed feedback during the preparation of the final report and the final presentation.

## 3.3. Client advisor

The Client adviser for this project is **Jan-Willem Manenschijn**. He is the Founder of Popup-Escape: a company that creates personalized escape rooms for events. The Client is the real-world stakeholder, who commissions the team to develop a software solution that addresses a specific problem. The Client provides the team with office space and gives the team the resources to user test the implemented product during its development.

## 3.4. Team

The team consists of 4 members: **Timo van Asten, Jael Lopez Kuchlin, Bernard Bot and Björn Ho**. The team will implement a software solution for the problem posed by the client. The team will also create a report on the whole project.

The team members have been assigned different roles within the team, as shown in table 3.1

Table 3.1: Team roles

| Roles | Member |
|---|---|
| Team Leader | Bernard Bot |
| Lead Programmer | Bernard Bot |
| Head of Communications | Timo van Asten |
| Management | Jael Lopez Kuchlin |
| Editor | Björn Ho |

**Team Leader:** The team leader is responsible for maintaining the proper course of the project. This team member insures that all requirements are met and the other team members perform their tasks correctly.

**Lead Programmer:** This team member is responsible for everything that happens with the codebase. The lead code makes sure that the code is of good quality: well documented, well tested and reviewed. The lead programmer evaluates all pull request and decides if they need to be corrected or merged. The weekly sprint retrospectives will be lead by this team member, especially the code review and the update of the sprint backlog.

**Head of Communications:** This team member is responsible for all communication, most notably with the TU Coach and the Client adviser. This person will also take the lead in creating the presentation and presenting it. The use study will be lead by this team member.

**Management:** This team member prepares the weekly agendas, minutes and schedules the meetings with the TU Coach and the Client Supervisor. The sprint retrospective meeting will be prepared together with the Lead programmer. This team member is in charge of managing and keeping a complete overview of all products and deliverables and their deadlines. User testing will be organized and evaluated primarily by this team member.

**Editor:** This team member is responsible for checking if all the requirements that are specified in the general guide is met in the report and also checking grammar, spelling and layout. All written documents that are delivered should be analysed by this team member. In the weekly sprint retrospectives this member will discuss the changes of the report.

## 3.5. Communications

Table 3.2: Communication between team and client or coach.

| Communication type | Description | Frequency | Format | Participants | Deliverable |
|---|---|---|---|---|---|
| Status report | Meetings with supervisor | Bi-weekly | In person | Supervisor and team | Status report and summary of work done |
| Status report | Meetings with client | Weekly | In person | Client and team | Status report and summary of work done |
| Meetings with supervisor | Sprints | Weekly | In person | All team members | Sprint retrospective Sprint backlog |

## 3.6. Project team directory

Table 3.3: Directory

| Name | Title | Email | Phone | Studentnr. |
|---|---|---|---|---|
| Willem-Paul Brinkman | TU Coach | w.p.brinkman@tudelft.nl | - | - |
| Jan-Willem Manenschijn | Client advisor | jw@manenschijn.com | - | - |
| Bernard Bot | Team leader | b.r.a.bot@student.tudelft.nl | - | - |
| Jael Lopez Kuchlin | Team member | j.r.r.lopezkuchlin@student.tudelft.nl | - | - |
| Björn Ho | Team member | b.i.y.l.ho@student.tudelft.nl | - | - |
| Timo van Asten | Team member | t.vanasten@student.tudelft.nl | - | - |

# 4

# Project Timeline

## 4.1. Deliverables

**These are the most important deliverables of the project:**

- Project Plan
- Research Report
- Final Report
- Presentation
- SIG 2x

## 4.2. Schedule

The full schedule of the entire project can be found on the next page. This schedule shows all deadlines and major tasks per week.

# Schedule

| **Sprints** | **Deadline** | **Planning**<br>*Work days are from 09:30-17:00* | **Tasks** |
|---|---|---|---|
| Week 1<br>*23/04/18 - 27/04/18* | 25/04: Project plan | 25/04: 10 AM meeting | - Outline final & research report<br>- Find research papers<br>- Mail Bachelor coördinator concerning approval<br>- Deliver Project Plan<br>- Set up development environment (first git repo push)<br>- Set up work space |
| Week 2<br>*30/04/18 - 04/05/18* | 04/05: Research report | | - Review research report<br>- Write research report<br>- Deliver research report<br>- Create sprint backlog from research report<br>- Set up (weekly?) user test |
| Week 3<br>*07/05/18 - 11/05/18* | | | - Deliver first prototype |
| Week 4<br>*14/05/18 - 18/05/18* | | 16/05: Timo out of office<br>18: Jael out of office<br>18: Bernard out of office | - Implement at least half of must haves |
| Week 5<br>*21/05/18 - 25/05/18* | | | - Implement all must haves |
| Week 6<br>*28/05/18 - 01/06/18* | 01/06: SIG 1st submission<br><br>28/05: Minimal viable product | 30/05: 1st field study and expert review | - mail d.bijlsma@sig.eu for exact date<br>- Central code review before SIG submission<br>- Implement at least half of should haves |
| Week 7<br>*04/06/18 - 08/06/18* | | 09/06: Large field study | - Implement all should haves<br>- Implement feedback from SIG |
| Week 8<br>*11/06/18 - 15/06/18* | | 11/06-16/06: Timo out of office | - Implement part of could haves<br>- Write outline presentation |
| Week 9<br>*18/06/18 - 22/06/18* | 22/06: SIG 2nd submission<br><br>22/06: 1st submission resolved | 21/06: 2nd field study | - Final code review<br>- Implement more could haves<br>- Bachelor Project Info sheet<br>- Final version final report<br>- Review final report |
| Week 10<br>*25/06/18 - 29/06/18* | 25/06: Final report | | - Deliver final report<br>- Final version presentation<br>- Practice presenting (time, wording, etc.) |
| Week 11<br>*02/07/18 - 06/07/18* | 2/07 - 06/07: Presentation (30 minutes, incl. demo) | | - Deliver presentation |

### 4.2.1. Tasks

Table 4.1: Weekly tasks of the team.

| Weekly tasks | Description |
|---|---|
| Individual retrospective | Short evaluation (what went right/wrong?) |
| Communal retrospective | Code review, report review, update sprint backlog |
| User testing | Use research, surveys, expert reviews |
| Final report | Update sections |
| Update planning | Weekly tasks |
| Schedule meeting TU Coach (bi-weekly) | Agenda |
| Schedule meeting Client Supervisor | Agenda |