# Project Pitch

**Team 99**

VU VRIJE UNIVERSITEIT AMSTERDAM

# Team

| Name | Student number | Email |
|---|---|---|
| Bence Matajsz | 2762173 | b.matajsz@student.vu.nl |
| Thijmen Verlaan | 2769118 | t.r.verlaan@student.vu.nl |
| Ana-Maria Musca | 2757719 | a.musca@student.vu.nl |
| Yasin Karyagdi | 2745681 | y.k.karyaaedi@student.vu.nl |
|  |  |  |

# Overview

For this project, we are going to design and implement the software for a file archiver. This file archiver can be interacted with through a command-line interface, which will support a handful of commands. The main user of this service will be an individual with the need to efficiently create a single archive representing multiple files.

The file archiver will adhere to the following functionalities:

- Compressing and decompressing archives with or without the usage of a password
- Listing the files contained in an archive
- Different compression methods
- Customisable configurations to be used for the creation of archives

# Feature 1

**Name**: Add files/folders to a new archive

**Description:**

This application should give the user the possibility to collect files and/or folders into a new archive by compressing them.

In order to use this feature, the user has to provide the name of the archive and the list of files and/or folders. Optionally the user can provide the type of archive they want. The default type will be ZIP.

The command line structure will be as follows: **create <name> <file list> [options]**, where the options field is an optional field that contains the flags and their corresponding info.

Whereby the implemented flags with be -f indicating the format preferred by the user, the -p flag indicates the password and the -c flag indicates the configurations.

Example:
create name file1 file2 folder1

**Champion:** Ana-Maria Musca

# Feature 2

**Name**: Extract contents

**Description:**

The system should allow users to extract all contents of an archive, into a user-defined folder.

If the user-defined folder already exists in the current directory, the system should not let the operation execute.

If the user does not define a folder to extract into, the system should create a new folder with the same name as the archive, in the current working directory.

The command line structure will be as follows:  **extract <name> [directory name]**, where the directory name field is optional. If password is required, the program will ask the user to enter the password.

Examples:
extract arch1

**Champion:** Bence Matajsz

# Feature 3

**Name**: Password-based encryption

**Description:**

The application should allow both password-based encryption and decryption.

The application should allow the user to provide a password to be used for the encryption and decryption of archives.

This feature should introduce the **-p <password>** option, allowing the user to create (feature 1) an archive using a particular password. If a password is not specified, the creation is done without the use of one. If a password is set when extracting (feature 2), the user will be prompted to provide it. If the user cannot guess the password after three tries, the execution of this command is canceled. If a password is not set when extracting, functionality remains as is.

This feature should use the Advanced Encryption Standard (AES) to be found in Java's standard library.

Example:

```
create myzip file.txt -p password123
```

**Champion:** Thijmen Verlaan

# Feature 4

**Name**: Explore archive content without extracting

**Description:**

The application should allow the user to explore the archive content, this is done without extracting the archive. By exploration we mean the listing of the content of the archive. The user should not be able to open the files contained in the archive, so when the **list** command is used the system will only return a list of the files (and their corresponding path) contained in the archive.

In order to use this feature the user has to use the **list** command in combination with name of the archive they want to list.

The command line structure will be as follows: **list <name>**, where the name should be the name of an archive of a valid format type. A valid format type is a format type which has been implemented in the system.

Examples:
list arch1

**Champion:** Yasin Karyagdi

# Feature 5

**Name**: Multiple compression formats

**Description:**

The application should allow multiple compression formats to be used when creating (feature 1) archives. These compression formats include ZIP and tar (using bzip2).

This feature should introduce the **-f <format>** option, allowing the user to create archives using a particular compression format. If no particular format is specified on creation, ZIP will be used.

For extraction (feature 2), the file extension is used to determine the compression format to be used for decompression.

Example:
create mytar file.txt -f tar

**Champion:** Thijmen Verlaan

# Feature 6

**Name**: Support of basic configurations for compression formats

**Description:**

The system shall allow the user to select the level (size) of compression when compressing a files.

Levels of configurations: none, low, medium, high

The user can use the following flag to configure compression, where **<level>** is the level of compression: **-c <level>**

If the user does not define a compression level, medium compression is used as default.

Example:
create myzip arg1.txt -c high

**Champion:** Bence Matajsz

# Feature 7

**Name**: Analyse compression formats

**Description:**

The application should allow users to submit a selection of files and directories and be shown a table with statistics regarding the compression process of this selection per implemented compression format using the default configuration level (feature 6). These statistics should include (1) the size of the compressed archive, (2) the percentage of reduction in size compared to the original selection, and (3) the duration of compressing the files into an archive.

The application should provide users with a recommendation on which format, out of the implemented ones, to use. This recommendation is based on the reduction in size achieved by the compression methods.

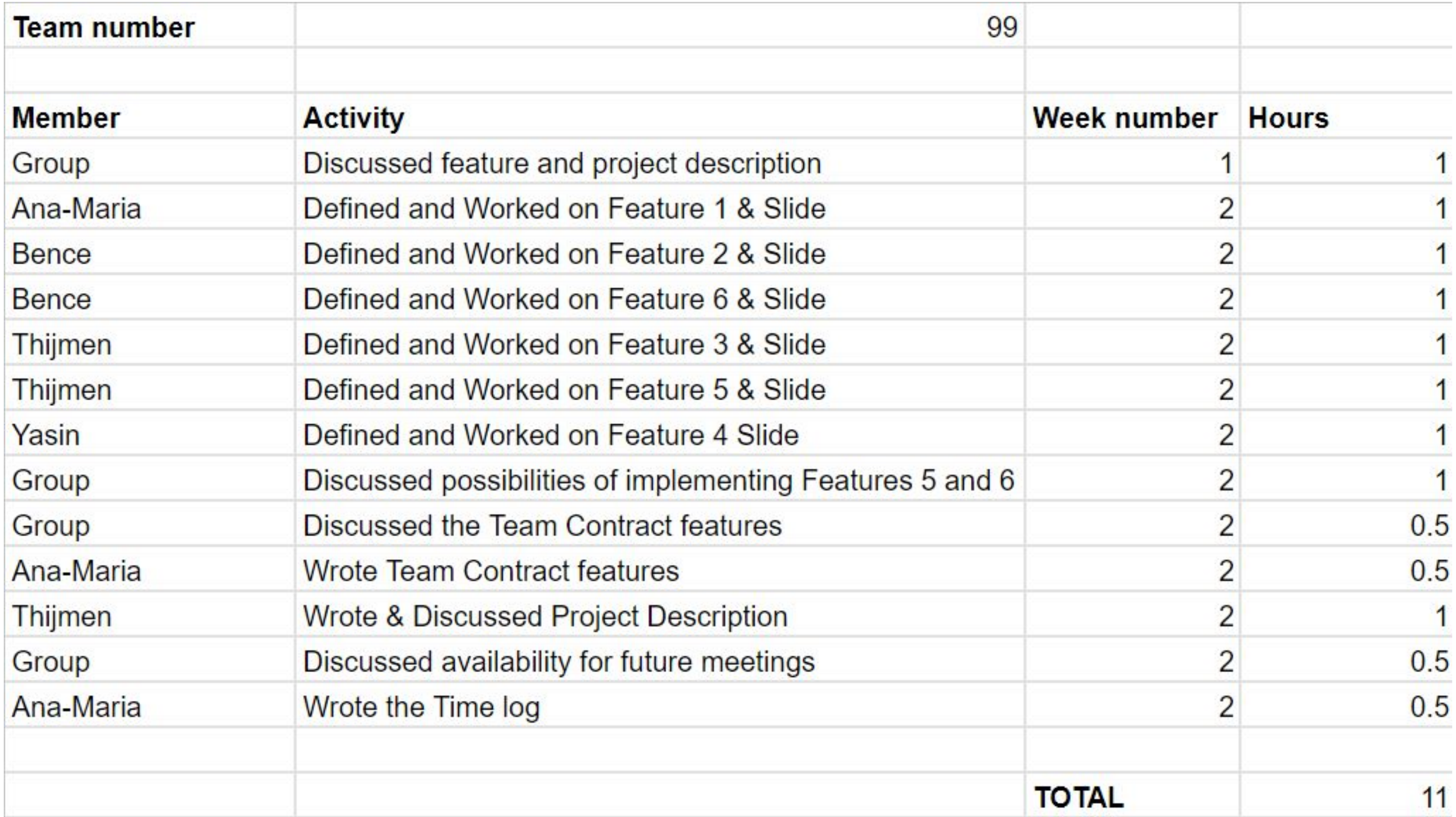
If the user does not provide a selection of files and directories, the recommendation and statistics will be based on an empty archive of each implemented compression format.

Example:

```
analyse cat.png dogs/ bird.jpg
```

**Champion:** Thijmen Verlaan

# Time log



| | | | |
|---|---|---|---|
| **Team number** | 99 | | |
| | | | |
| **Member** | **Activity** | **Week number** | **Hours** |
| Group | Discussed feature and project description | 1 | 1 |
| Ana-Maria | Defined and Worked on Feature 1 & Slide | 2 | 1 |
| Bence | Defined and Worked on Feature 2 & Slide | 2 | 1 |
| Bence | Defined and Worked on Feature 6 & Slide | 2 | 1 |
| Thijmen | Defined and Worked on Feature 3 & Slide | 2 | 1 |
| Thijmen | Defined and Worked on Feature 5 & Slide | 2 | 1 |
| Yasin | Defined and Worked on Feature 4 Slide | 2 | 1 |
| Group | Discussed possibilities of implementing Features 5 and 6 | 2 | 1 |
| Group | Discussed the Team Contract features | 2 | 0.5 |
| Ana-Maria | Wrote Team Contract features | 2 | 0.5 |
| Thijmen | Wrote & Discussed Project Description | 2 | 1 |
| Group | Discussed availability for future meetings | 2 | 0.5 |
| Ana-Maria | Wrote the Time log | 2 | 0.5 |
| | | | |
| | | **TOTAL** | 11 |

# Signed contract

Team Contract