## Task I: Requirements

*a) Determine all the actors involved in the system.*

**HUMAN**
- Creators
- Backers (Funders)
- Administrators
- Bank Companies
- Viewers

**NON-HUMAN**
- Database

*b) Determine as many significant functional requirements as you can, and express them as user stories*

- As a creator, I want to propose projects so that the project I am proposing gets a funding for it to function well.
- As a creator, I want to make a detailed budget of my project's cost so that I can set my funding goal.
- As a creator, I want to brainstorm some rewards to offer for my project so that my project may be advertised.
- As a creator, I want to plan out information to share so that I can earn more of my backers' trust.
- As a creator, I should include a video into my project launch so that the success rate for the project can statistically increase.
- As a creator, I want to limit reward shipping so that my budget is kept in its range and not exceed due to shipping fees for rewards.
- As a creator, I want to share my project before I launch it so that my project's information may be spread to advertise.
- As a creator, I want to list all the people involved in the project so that each person can get credibility.
- As a creator, I want to acknowledge all rights of media creators if I am using media so that I won't be implementing anything illegally and potentially be subjected to lawsuits.
- As a creator, I want to be able to have my project page to be displayed in various languages so that people of various cultures are able to read my project and its proposal.
- As a creator, I want to find ways to advertise my project so that money can be raised for my project.
- As a creator, I want to include an eye-catching image so that I can draw attention from viewers to read my project's proposal.
- As a creator, I want to have a cohesive and relatable project video so that the viewers can clearly understand who we are, what our project is, why it is important to us and why it should be important to others.
- As a creator, I want to utilise the maximum image size of 50MB so that my project may look attractive and still look graphical to appeal to the target audience.
- As a creator, I want to manage the time taken for the project to build and finish so that my project doesn't stay on a standstill and continue progressing to its final goal.
- As a creator, I want my project to be reviewed so that I can get feedback and improve on the project's layout to draw in more people for my project's purposes.

- As a creator, I want to track my project's progress so that I know how much funding the project has gotten and how progressive the project has gone along.
- As a creator, I may want to edit my project after launching so that I can update old information and provide viewers with more recent details.
- As a creator, I want to answer questions from backers and non-backers so that my information for the project isn't misleading.
- As a creator, I want to utilise Spotlight to advertise my finished project so that I can inform people about news and updates on the concluded project.
- As a creator, I want to retry my project if it was unsuccessful so that my project can be refunded and spread information to the wider community.
- As a creator, I want to register my company so that I have a legal entity to receive funds.
- As a creator, I want to provide my bank account details so that the funding raised for my project may be sent to the designated account.
- As an administrator, I want to maintain the current system to its maximum capacity so that people are able to use system to endorse their projects.
- As a backer, I want to fund various projects so that the projects can provide their services with the funding raised.
- As a backer, I want to find interesting projects to back so that the project may benefit me later on if it succeeds.
- As a backer, I want to pledge to a project so that the project may be funded for its services.
- As a backer, I want to link my Facebook account to this service so that I may be able to access projects by an easier means and get frequent updates.
- As a backer, I want to keep my account secure so that I don't encounter any false pledging or backing of various projects that I didn't agree to.
- As an administrator, I want to include inclusivity issues into the layout of the system so that all people whether they have disabilities or cultural difficulties can use the system.
- As an administrator, I want to stabilise the system in terms of functionality so that people using the system may use it without any troubleshooting.
- As a bank company, I want to assist the program in interchanging money to the creators and other people so that the bank may be creditable to advantages such as funds for using the service.
- As a viewer, I want to look for various projects so that I may be able to become a backer and fund those projects.

*c) Pick four user stories, substantively different from each other, and develop each one into a complete use case*

1.As a backer, I want to find interesting projects to back so that the project may benefit me later on if it succeeds.

**Goal:** Fund a project
**Primary Actor:** Backer
**Secondary Actor:** Creator
**Trigger:**Backer selects the 'Fund Project' option.
**Precondition:** Project must be initialised.
**Flow of Events:**
1. Backer inputs the amount of money that they are willing to fund towards the project.
2. Backer selects the 'next' option.
3. Backer chooses a payment method
4. Backer inputs payment details
5. Backer selects the 'fund' option.

6. Backer is notified through e-mail that they have successfully funded the project.
**Extensions:**
6A1. Backer is notified on screen that the payment details are invalid.
6A2. The system prompts the backer to re-input their payment details.

2.As a creator, I want to propose projects so that the project I am proposing gets a funding for it to function well.

**Goal:** Propose a project
**Primary Actor:** Creators
**Secondary Actor:** Backers
**Trigger:**Creator selects the 'Propose project'
**Precondition:** Project must be completely planned out.
**Flow of Events:**
1. Creator inputs information about their project into the fields provided.
2. Creator when finished selects the 'next' option.
3. Creator is prompt to input their bank account details for funding transfer
4. Creator inputs their bank account details
5. Creator selects the 'next' option.
6. Creator is notified on screen of the confirmation of the lodging of their project
7. Creator selects the 'lodge' option
8. Creator is notified through e-mail that they have proposed a project.
**Extensions:**
5A1. Creator is notified on screen that their bank account details are invalid.
5A2. The system prompts the creator to re-input details of a valid bank account.

3. As a creator, I want my project to be reviewed so that I can get feedback and improve on the project's layout to draw in more people for my project's purposes.

**Goal:** Get the project reviewed
**Primary Actor:** Administrators
**Secondary Actor:** Creators
**Trigger:** Creator sends the project through e-mail to administrator for review.
**Precondition:** The project has been completed for reviewing.
**Flow of Events:**
1. Administrator receives project through e-mail.
2. Administrator redirects the e-mail to another department for reviewing.
3. Administrator, after the project has been reviewed, receives an e-mail of the reviewed project.
4. Administrator e-mails the creator about the reviewed project with the project as an attachment.
5. Creator receives the reviewed project.
6. Creator edits the project to fit with the reviewed project.
**Extensions:**
2A1. The administrator reviews the project.
2A2. Administrator e-mails the creator with the reviewed project attachment.

4.As a backer, I want to link my Facebook account to this service so that I may be able to access projects by an easier means and get frequent updates.

**Goal:** Link my Facebook account to the project's account.
**Primary Actor:** Creator
**Secondary Actor:** Database

**Trigger:** Creator selects the 'edit' option to access his account details.
**Precondition:** The creator must have a functioning Facebook account.
**Flow of Events:**
1. Creator finds the field to input his Facebook account details.
2. Creator inputs his Facebook account details.
3. The system sends the information to the database for verification.
4. The system prompts to the user that the verification was successful and that the account is linked to Facebook.

**Extensions:**
4A1. The system prompts to the creator that the verification was unsuccessful and that the account is not valid
4A2. The system prompts the creator to re-input valid details of their Facebook account details.

*d) For each of your use cases, come up with rough plan for testing that functionality. In a paragraph of plain English, explain how you would verify whether the final system correctly implements the use case.*

1. To test this functionality within the final system, I would attempt to fund with a small amount of money to see if the project gets funding. I would also utilise a pseudo e-mail address and link it to the backer's account to see, when they fund a project, if I receive an e-mail notifying that they have attempted to fund a project with a certain amount of money.

2. To test whether proposing a project was successful, I would attempt to create a project with fake information for each of the require fields the system needs. I would also create a pseudo e-mail address for the system to send me an e-mail notifying me if I successfully proposed a project.

3. To test whether this functionality works, I would create two pseudo e-mail addresses and select one of them to be the creator's email and one to be the system e-mail. I would then create a fake project with fake information in the fields that are required. I would link the system's email to the link that prompts the user to select when they need to review a project. After the project if completed with fake information, I will test if the project is sent off to the system's e-mail through the 'review' link and see if it is received. If it is received, I would make a few changes and resend the project back to the creator's email.

4. To test this functionality, I would create a pseudo Facebook account and a project account for link testing. I would then link my Facebook account through the project account settings. This will then link my Facebook account with the project's system. I would then see if the account can be accessed through Facebook through its login screen. If it is successful, then the link for the Facebook account was successful.

## TASK II: Testing

*a) Provide a partial design of each of the three production code submodules. You do not need to provide algorithms. For each submodule, state its name, its imports (names and datatypes) and exports (where applicable), and its database inputs and outputs. Also describe briefly what the submodule does with its imports and inputs and what it gives back in its export and outputs.*

Submodule **exceedAmount**
Imports: **backerPledge (double), currentAmount(double), totalAmount (double)**
Exports: **state (String)**

Calculates if the amount that the backer pledged (backerPledge) added towards the current amount that the project has (currentAmount) and see if it will exceed the funding goal the project has (totalAmount). It will then return a string (state) stating whether the funding exceeded the funding goal. All imports are non-negative real numbers. If currentAmount is less than 0, the value of 0 is used instead. If backerPledge or totalAmount is less than 0, the value of 1 is used instead. The export state will either display "Exceeded" or "Not Exceeded" or "Equal". This function will then return to the user a string to signify if the pledge exceeded the amount the project requires.

Submodule **search**
Imports: **searchWord (String), projectDescription (String)**
Exports: **found (String)**

Searches the description of all projects (projectDescription) for the word the user has given for search (searchWord). If searchWord is amongst projectDescription, then this function will output to the user if the keyword was amongst any of the project descriptions by outputting either "Found" or "Not Found".

Submodule **fundingPerDay**
Imports: **currentAmount (double), totalAmount (double), daysLeft (integer)**
Exports: **amountPerDay (double)**

Calculates the amount needed per day (amountPerDay) to reach the funding goal (totalAmount) with the amount of days remaining until the project is finished (daysLeft), using the current amount of funding the project has (currentAmount). All imports are non-negative real numbers. If the currentAmount is less than 0, the value of 0 is used instead. If the totalAmount is less than 0, the value of 1 is used instead. If daysLeft is less than 0, the value of 0 is used instead. If there are no days remaining, the function will output a value of 0. This function will return a double specifying how much is needed to be raised with the remaining days left to reach the funding goal and 0 otherwise.

*b) Design test cases for each production code submodule, using equivalence partitioning.*

Submodule **exceedAmount**
Imports: **backerPledge (double), currentAmount(double), totalAmount (double)**
Exports: **state (String)**

Calculates if the amount that the backer pledged (backerPledge) added towards the current amount that the project has (currentAmount) and see if it will exceed the funding goal the project has (totalAmount). It will then return a string (state) stating whether the funding exceeded the funding goal. All imports are non-negative real numbers. If currentAmount is less than 0, the value of 0 is used instead. If backerPledge or totalAmount is less than 0, the value of 1 is used instead. The export state will either display “Exceeded” or “Not Exceeded” or “Equal”. This function will then return to the user a string to signify if the pledge exceeded the amount the project requires.

- **backerPledge< 0**
  IMPORT: backerPledge = -3.0, currentAmount = 2.0, totalAmount = 5.0
  EXPORT: state = “Not Exceeded”

- **currentAmount< 0**
  IMPORT: backerPledge = 1.0, currentAmount = -1.0, totalAmount = 5.0
  EXPORT: state = “Not Exceeded”

- **totalAmount< 0**
  IMPORT: backerPledge = 2.0, currentAmount = 3.0, totalAmount = -1.0
  EXPORT: state = “Exceeded”

- **backerPledge< 0, currentAmount< 0**
  IMPORT: backerPledge = -2.0, currentAmount = -1.0, totalAmount = 5.0
  EXPORT: state = “Not Exceeded”

- **backerPledge< 0, totalAmount< 0**
  IMPORT: backerPledge = -1.0, currentAmount = 2.0, totalAmount = -5.0
  EXPORT: state = “Exceeded”

- **currentAmount< 0, totalAmount< 0**
  IMPORT: backerPledge = 1.0, currentAmount = -2.0, totalAmount = -5.0
  EXPORT: state = “Equal”

- **backerPledge< 0, currentAmount< 0, totalAmount< 0**
  IMPORT: backerPledge = -1.0, currentAmount = -2.0, totalAmount = -5.0
  EXPORT: state = “Equal”

- **backerPledge> 0, currentAmount> 0, totalAmount> 0**
  IMPORT: backerPledge = 1.0, currentAmount = 2.0, totalAmount = 5.0
  EXPORT: state = “Not Exceeded”

Submodule **search**
Imports: **searchWord (String), projectDescription (String)**
Exports: **found (String)**

Searches the description of all projects (projectDescription) for the word the user has given for search (searchWord). If searchWord is amongst projectDescription, then this function will output to the user if the keyword was amongst any of the project descriptions by outputting either "Found" or "Not Found".

- **projectDescription contains searchWord**
  IMPORT: searchWord = "Projection", projectDescription = "The Arc of Projection"
  EXPORT: found = "Found"

- **projectDescription doesn't contain searchWord**
  IMPORT: searchWord = "Orange", projectDescription = "The Plane's Wings"
  EXPORT: found = "Not Found"

- **projectDescriptionis the same assearchWord**
  IMPORT: searchWord = "Orange is orange", projectDescription = "Orange is orange"
  EXPORT: found = "Found"

Submodule **fundingPerDay**
Imports: **currentAmount (double), totalAmount (double), daysLeft (integer)**
Exports: **amountPerDay (double)**

Calculates the amount needed per day (amountPerDay) to reach the funding goal (totalAmount) with the amount of days remaining until the project is finished (daysLeft), using the current amount of funding the project has (currentAmount). All imports are non-negative real numbers. If the currentAmount is less than 0, the value of 0 is used instead. If the totalAmount is less than 0, the value of 1 is used instead. If daysLeft is less than 0, the value of 0 is used instead. If there are no days remaining, the function will output a value of 0. This function will return a double specifying how much is needed to be raised with the remaining days left to reach the funding goal and 0 otherwise.

- **currentAmount< 0**
  IMPORT: currentAmount = -1.0, totalAmount = 5.0, daysLeft = 5
  EXPORT: amountPerDay = 1.0

- **totalAmount< 0**
  IMPORT: currentAmount = 0.0, totalAmount = -1.0, daysLeft = 10
  EXPORT: amountPerDay = 0.1

- **daysLeft< 0**
  IMPORT: currentAmount = 2.0, totalAmount = 5.0, daysLeft = -2
  EXPORT: amountPerDay = 0.0

- **currentAmount< 0, totalAmount< 0**
  IMPORT: currentAmount = -1.0, totalAmount = -2.0, daysLeft = 10
  EXPORT: amountPerDay = 0.1

- **currentAmount< 0, daysLeft< 0**
  IMPORT: currentAmount = -2.0, totalAmount = 12.0, daysLeft = -2
  EXPORT: amountPerDay = 0.0

- **totalAmount< 0, daysLeft< 0**
  IMPORT: currentAmount = 2.0, totalAmount = -5.0, daysLeft = -5
  EXPORT: amountPerDay = 0.0

- **currentAmount< 0, totalAmount< 0, daysLeft< 0**
  IMPORT: currentAmount = -5.0, totalAmount = -2.0, daysLeft = -3
  EXPORT: amountPerDay = 0.0

- **currentAmount> 0, totalAmount> 0, daysLeft> 0**
  IMPORT: currentAmount = 3.0, totalAmount = 10.0, daysLeft = 14
  EXPORT: amountPerDay = 0.5

c) Determine what set of test-support submodules (e.g. createFakeDB) you will need. Provide their names, imports, exports, inputs and outputs (whichever are applicable), and a short description

Submodule **testExceedAmount**
Imports: **none**
Exports: **none**

This function will assert the test cases from part b) for exceedAmount() and equate them to the expected value.

Submodule **testSearch**
Imports: **none**
Exports: **none**

This function will assert the test cases from part b) for search() and equate them to the expected value.

Submodule **testFundingPerDay**
Imports: **none**
Exports: **none**

This function will assert the test cases from part b) for fundingPerDay() and equate them to the expected value.

(d) Implement your test cases using JUnit (using test-support submodules as needed)

```
import io.*;
public class TestCode
{
	public static String exceedAmount(double backerPledge, double currentAmount, double
totalAmount)
	{
		double tempVal;
		String state = "";

		if (backerPledge < 0.0)
		{
			backerPledge = 1.0;
		}
		if (currentAmount < 0.0)
		{
			currentAmount = 0.0;
		}
		if (totalAmount < 0.0)
		{
			totalAmount = 1.0;
		}

		tempVal = backerPledge + currentAmount;
		if (tempVal < totalAmount)
		{
			state = "Not Exceeded";
		}
		else if (tempVal > totalAmount)
			{
				state = "Exceeded";
			}
			else if (tempVal == totalAmount)
				{
					state = "Equal";
				}

		return state;
	}

	public static String search(String searchWord, String projectDescription)
	{
		String found;

		if (projectDescription.toLowerCase().contains(searchWord.toLowerCase()))
		{
			found = "Found";
		}
		else
		{
			found = "Not Found";
		}

		return found;
	}

	public static double fundingPerDay(double currentAmount, double totalAmount, int
daysLeft)
	{
		double amountPerDay, tempVal;

		if (currentAmount < 0.0)
		{
			currentAmount = 0.0;
		}
		if (totalAmount < 0.0)
		{
			totalAmount = 1.0;
		}
		if (daysLeft < 0)
		{
			daysLeft = 0;
		}

		tempVal = totalAmount - currentAmount;
		if (daysLeft == 0)
		{
			amountPerDay = 0;
		}
		else
		{
			amountPerDay = tempVal / (double)daysLeft;
		}
		return amountPerDay;
	}
}
```

-----------------------------------------------------------------------------------------------------------------------------

```
import org.junit.*;
import org.junit.runner.RunWith;
import org.junit.runners.JUnit4;
import static org.junit.Assert.*;

@RunWith(JUnit4.class)
public class TestCodeTest
{
	@Test
	public void testExceedAmount()
	{
	assertEquals("Backer Pledge < 0", "Not Exceeded", TestCode.exceedAmount(-3.0, 2.0, 5.0));
```

```
		assertEquals("Current Amount < 0", "Not Exceeded", TestCode.exceedAmount(1.0, -1.0,
5.0));
		assertEquals("Total Amount < 0", "Exceeded", TestCode.exceedAmount(2.0, 3.0, -1.0));
		assertEquals("Backer Pledge < 0 & Current Amount < 0", "Not Exceeded",
TestCode.exceedAmount(-2.0, -1.0, 5.0));
		assertEquals("Backer Pledge < 0 & Total Amount < 0", "Exceeded",
TestCode.exceedAmount(-1.0, 2.0, -5.0));
		assertEquals("Current Amount < 0 & Total Amount < 0", "Equal",
TestCode.exceedAmount(1.0, -2.0, -5.0));
		assertEquals("All Imports < 0", "Equal", TestCode.exceedAmount(-1.0, -2.0, -5.0));
		assertEquals("All Imports > 0", "Not Exceeded", TestCode.exceedAmount(1.0, 2.0, 5.0));
		}

		@Test
		public void testSearch()
		{
		assertEquals("Description contains Search Word", "Found",
TestCode.search("Projection","The Arc of Projection"));
		assertEquals("Description doesn't contain Search Word", "Not Found",
TestCode.search("Orange","The Plane's Wings"));
		assertEquals("Description is the same as the Search Word", "Found",
TestCode.search("Orange is orange","Orange is orange"));
		}

		@Test
		public void testFundingPerDay()
		{
		assertEquals("Current Amount < 0", 1.0, TestCode.fundingPerDay(-1.0, 5.0, 5), 0.0001);
		assertEquals("Total Amount < 0", 0.1, TestCode.fundingPerDay(0.0, -1.0, 10), 0.0001);
		assertEquals("Days Left < 0", 0.0, TestCode.fundingPerDay(2.0, 5.0, -2), 0.0001);
		assertEquals("Current Amount < 0 & Total Amount < 0", 0.1, TestCode.fundingPerDay(-1.0, -
2.0, 10), 0.0001);
		assertEquals("Current Amount < 0 & Days Left < 0", 0.0, TestCode.fundingPerDay(-2.0, 12.0,
-2), 0.0001);
		assertEquals("Total Amount < 0 & Days Left < 0", 0.0, TestCode.fundingPerDay(2.0, -5.0, -5),
0.0001);
		assertEquals("All Imports < 0", 0.0, TestCode.fundingPerDay(-5.0, -2.0, -3), 0.0001);
		assertEquals("All Imports > 0", 0.5, TestCode.fundingPerDay(3.0, 10.0, 14), 0.0001);
		}
}
```