Clickster 9000

Team Members and Roles

- **Project lead**: Ian Dykas
- **Art designers**: Shelnesha Taylor, Salena Youhana, Lauren Broski, Ian Dykas
- **Programmers**: Shelnesha Taylor, Ian Dykas, Salena Youhana, Mayuri Candagaddala, Lauren Broski
- **Testers**: Ian Dykas, Salena Youhana, Mayuri Candagaddala, Lauren Broski
- **Music designers**: Mayuri Candagaddala, Ian Dykas

Phase 1

Project Development

Problem Statement

People do not have the ultimate clicking game, so they need our game that will entertain the user constantly. On top of this, there are not enough clicking games out there, so we need to make another one. Our clicking game solves the issue of not knowing what clicking game to play and the case of boredom.

Objectives and Impact

As the school year is wrapping up, many high school and college students are dealing with stress of final exams, papers, and projects. We have developed a clicking game that will take the stress off of student's minds and give them a much needed break. Its simple design won't require users to think too much. Its short time frame ensures won't have users stuck for hours trying to complete the game.

Functional requirements

- Click screen, get points
- Use points in a shop
- Having an end goal instead of the game being infinite
- Ability to save progress in game
- Achievement system
- Milestone bonus

Nonfunctional Requirements

- Usability – everything working as intended without exploits in the game logic
- Scalability – we can add more features in updates over time
- Performance – make experience as smooth as possible
- Reliability and Recoverability - save system retrieving correct data

Target Environment

Java, non-specific operating system.

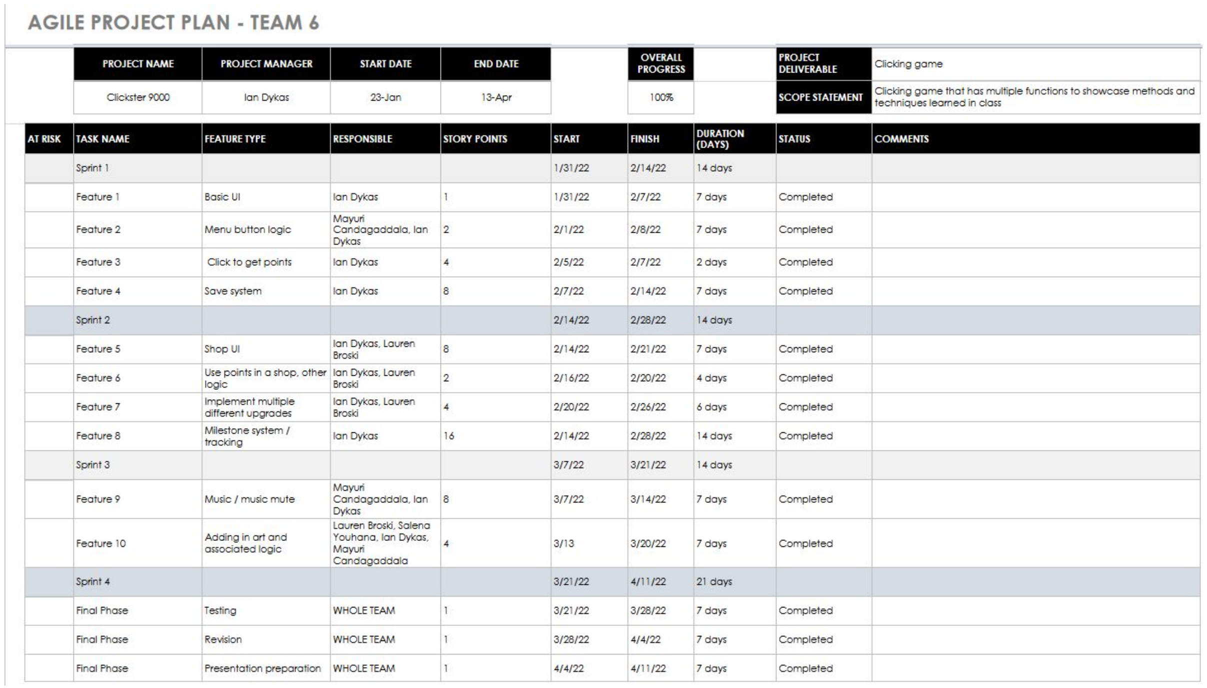
Technology and Tools

- We will be making use of Java and its many libraries for this game, targeting users on PC.
- Utilizing read and write Java functions to save a file.
- Basic algorithms for “shop items” that give players bonuses.
- A lot of user interface usage, the majority of the game is a user interface.
- Github for code collaboration
- Netbeans as our IDE
- Google sheets for project management

Process Model Used

We used the Scrum model since it details specific meeting times for sprints, which can helped us stay organized and on time with development. It will also allow us to stay on the same page when developing our application.

Project Schedule



AGILE PROJECT PLAN - TEAM 6

PROJECT NAME	PROJECT MANAGER	START DATE	END DATE	OVERALL PROGRESS
Clickster 9000	Ian Dykas	23-Jan	13-Apr	100%

PROJECT DELIVERABLE	Clicking game
SCOPE STATEMENT	Clicking game that has multiple functions to showcase methods and techniques learned in class

AT RISK	TASK NAME	FEATURE TYPE	RESPONSIBLE	STORY POINTS	START	FINISH	DURATION (DAYS)	STATUS	COMMENTS
	Sprint 1				1/31/22	2/14/22	14 days		
	Feature 1	Basic UI	Ian Dykas	1	1/31/22	2/7/22	7 days	Completed	
	Feature 2	Menu button logic	Mayuri Candagaddala, Ian Dykas	2	2/1/22	2/8/22	7 days	Completed	
	Feature 3	Click to get points	Ian Dykas	4	2/5/22	2/7/22	2 days	Completed	
	Feature 4	Save system	Ian Dykas	8	2/7/22	2/14/22	7 days	Completed	
	Sprint 2				2/14/22	2/28/22	14 days		
	Feature 5	Shop UI	Ian Dykas, Lauren Broski	8	2/14/22	2/21/22	7 days	Completed	
	Feature 6	Use points in a shop, other logic	Ian Dykas, Lauren Broski	2	2/16/22	2/20/22	4 days	Completed	
	Feature 7	Implement multiple different upgrades	Ian Dykas, Lauren Broski	4	2/20/22	2/26/22	6 days	Completed	
	Feature 8	Milestone system / tracking	Ian Dykas	16	2/14/22	2/28/22	14 days	Completed	
	Sprint 3				3/7/22	3/21/22	14 days		
	Feature 9	Music / music mute	Mayuri Candagaddala, Ian Dykas	8	3/7/22	3/14/22	7 days	Completed	
	Feature 10	Adding in art and associated logic	Lauren Broski, Salena Youhana, Ian Dykas, Mayuri Candagaddala	4	3/13	3/20/22	7 days	Completed	
	Sprint 4				3/21/22	4/11/22	21 days		
	Final Phase	Testing	WHOLE TEAM	1	3/21/22	3/28/22	7 days	Completed	
	Final Phase	Revision	WHOLE TEAM	1	3/28/22	4/4/22	7 days	Completed	
	Final Phase	Presentation preparation	WHOLE TEAM	1	4/4/22	4/11/22	7 days	Completed	

Staff Related Risk

Project: Clickster 9000
Risk type: Staff-related risk
Priority (1 low ... 5 critical): 2
Risk factor: Some group members might not be familiar with clicking games.
Probability: 50%
Impact: 3 days were taken by the team to familiarize themselves with different clicking games and the different capabilities that most available clicking games, on the market, provide for users.
Monitoring approach: The team shared with each other when they learned about the functionality of different clicking systems and felt comfortable enough to start programming.
Contingency plan: Thinking of different ways to understand the clicking game concept.
Estimated resources: 3 days at the start of development.

Tech Related Risk

Project: Clickster 9000
Risk type: Tech-related risk
Priority (1 low … 5 critical): 5
Risk factor: Making a game from scratch might be a new experience to some of us.
Probability: 70%
Impact: Development will slow down if everyone does not understand how to make a game.
Monitoring approach: Everyone on the team learned concepts to help development.
Contingency plan: Possible back-up game ideas if things do not go to plan.
Estimated resources: Multiple weeks of time to develop game concepts with possible varying levels of confusion and skill sets.

Project Management Related Risk

Project: Clickster 9000
Risk type: Project Management-related risk
Priority (1 low … 5 critical): 3
Risk factor: Making sure that everything is in order and working right
Probability: 40%
Impact: Without proper management, the project could end up not working as intended, being inconsistent, and/or not completed at all.
Monitoring approach: Everyone on the team learned about managing their time and work flow
Contingency plan: Communication to get everyone back on track and on the same page.
Estimated resources: Multiple weeks to manage and work on developing the game while working on other work.

Completeness Related Risk

Project: Clickster 9000
Risk type: Completeness-related risk
Priority (1 low … 5 critical): 4
Risk factor: Some extra features might not get fully completed, or we don’t put enough features.
Probability: 60%
Impact: The game will not have enough features or may not be fully complete when a user tries to play it.
Monitoring approach: Making sure all the requirements or features are done by a certain date.
Contingency plan: Have new ideas for additional features if some do not work out. Make a schedule so we know when everything should be completed.
Estimated resources: Multiple weeks to complete all the requirements for the game and follow the schedule so we don’t get held back.

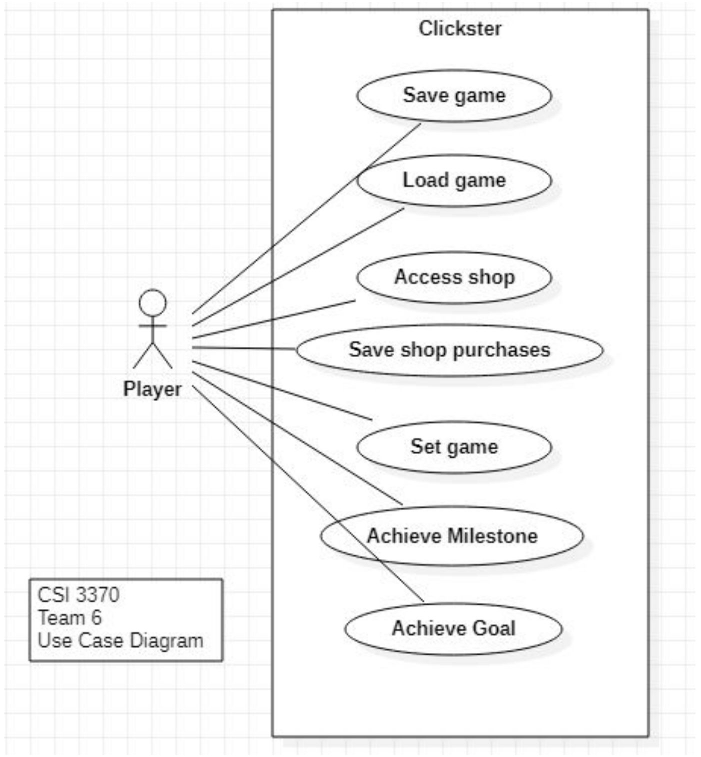
Performance Related Risk

Project: Clickster 9000
Risk type: Performance-related risk
Priority (1 low ... 5 critical): 2
Risk factor: Lag might throw the game into non-synchronization or another technical difficulty may shut down the user's progress entirely.
Probability: 10%
Impact: Users may experience frustration if the application is experiencing issues at a rate that it causes a lot of disruptions. As a result, users may choose to entirely leave our application and find something.
Monitoring approach: Look at the rate at which problems are being reported by users**.**
Contingency plan: Depending on what the root cause for the problem is, we would run diagnostic tests and try to troubleshoot the issue. More than likely at that time, the system will go offline.
Estimated resources: The development team would come together and troubleshoot the issue ideally within an hour or two. Depending on the level of complexity that is required to fix the system, it may take up to a day.

Phase 2

Design and Implementation

Use Case Diagram



Save the Game

Scope: Clickster
Primary Actor: Player
Stakeholders and Interests:

Player: Wants to keep their progress and have the ability to restart from their last saved spot.

Developer: Want to create a functional application that makes sure users will want to continue playing, that users will have a satisfactory experience.

Precondition:

Players must have the game running

Success Guarantee:

The game is saved and can be accessible at any point in the future.

Main Success Guarantee:

1. User opens the game
2. System loads the game
3. User plays through the game
4. System tracks the user
5. User clicks the save button
6. System initiates saving the file
7. System saves their progress
8. User exits the game.

Extensions:

*a. At any time, the System fails to save the user's progress:

1. User saves the game
2. System initiates the saving progress
3. System fails the saving process
4. System initiates an error, explaining that the file could not be saved
5. User retires save button
6. System initiates saving the file
7. System saves their progress.

*b. At any time, the User fails to save the system:

1. User exits the game
2. System does not save the user's progress

Special Requirements:

Save button occupies 20% of screen, when clicked message confirming the save appears in the middle of the display Saving Progress happens within 30 seconds after the user confirms their decision, or exits immediately if they decline.

Technology and Data Variations List:

- Mouse to physically play the game
- Monitor to show the user the progress they are making on each level and display the buttons the user needs to interact with

Frequency of Occurrence:

As often as the user wishes to save their progress, it could be a continuous function. More realistically it would be when the user needs to leave the game for an extended period of a time.

Open Issues:

- What happens if the system crashes before the user can save?
- How to save locally?
- What happens if the last save file is corrupted?

Access Shop

Scope: Clickster
Primary Actor: Player
Stakeholders and Interests:

Player: Keeps the player wanting to play the game more, earning more points to buy items.

Developer: After a certain amount of points, allows the user to shop for upgrades.

Precondition:

Players must get enough points to be able to shop.

Success Guarantee: Player achieves the goal of earning points, will be able to use points to shop and upgrade.

Main Success Guarantee:

1. Player starts the game.
2. System loads the system .
3. Player clicks throughout the game.
4. System gives points to the user based on each click.
5. System shows the user the information on the total number of points they got.
6. Player visits the shop to see what they can buy using said points.
7. System loads the shop.
8. Players can purchase items based on the number of points.
9. System provides the user with the upgrade they purchased.
10. Player continues clicking, if they visit the shop in the middle of the game.

Extensions:

*a. At any time, player wants to see the store in the middle of the game:
1. Player accesses the shop.
2. System displays the shop to the user.
3. The Player can select a shop item.

Special Requirements:

- Shop button should be visible on the main screen for easy access.
- When pausing the game, there should be the store button available as well as the other buttons that are required.

Technology & Data Variations List:

- Mouse to select store button
- Input of users points on screen while playing.

Frequency of Occurrence:

When the player earns points.

Open Issues:

- What happens when after finishing the game there are still points leftover?
- What type of items and upgrades will be available?
- Are there other ways to earn points?
- Depending on the item/upgrade, how long will it last? (like a power-up boost or a freeze-time)
- How to determine the worth of each item and upgrade

Achieve Milestone

Scope: Clickster
Primary Actor: Player
Stakeholders and Interests:
Player: Keeps the player engaged by reaching goals with incentives in the game.

Precondition:
Players must make enough progress to be able to achieve a milestone.

Main Success Guarantee:
Player achieves a milestone. System rewards player with an incentive from milestones.

Main Success Scenario:
1. Player starts the game.
2. System loads the game.
3. Player makes progress, by clicking in the game.
4. System tracks how many clicks the player is making.
5. Player reaches a set milestone.
6. System rewards the player with incentive from a milestone.
7. System notifies the player of milestone achievement.
8. Player is rewarded from the milestone.

Frequency of Occurrence:
When the Player makes enough progress to achieve the milestone.

Extensions:
*a. At any time, System does not register milestone completed by Player:
1. System checks progress on the next startup of the application.
2. System provides the skipped milestone incentive to the Player.
3. System notifies the Player of the milestone being achieved.
4. Player obtains incentive from the milestone menu.

Special Requirements:
A description for a milestone and its incentive must be made clear to the user to be able to achieve it. When the user achieves a milestone, a quick response must be given to the user to let them know that it was achieved.

Technology and Data Variations List:
Input of user’s progress from the save system to check for milestones achieved on start up if needed to do so. Mouse to click to achieve milestones.

Open Issues:
- What kind of milestones will the Player have to achieve?
- How will the System’s performance be affected by the milestone system?

Set Game

Scope: Clickster
Primary Actor: Player
Stakeholders and Interests:

Player: Can modify certain aspects of the game to help with their experience.

Developers: Less complaints about user experience.

Precondition:

Player must have the game launched.

Success Guarantee:

Player's selected settings are saved. System shows the impact from selected settings.

Main Success Scenario:

1. Player launches the game.
2. System provides the main screen to Player.
3. Player selects the Settings option on the main screen.
4. System provides Settings screen to Player.
5. Player chooses settings that they want.
6. System reflects chosen settings.

Extensions:

*a. At any time, Player wants to revert Settings back to default:

1. Player selects the Settings option on the main screen.
2. System provides Settings screen to Player.
3. Player selects a default settings option.
4. System reflects settings that the game started as.

Special Requirements:

Settings menu must be visibly clear as to what options are available, along with each option's functions described clearly.

Technology and Data Variations List:

- Output of chosen settings to be used as a preset for when the Player decides to play the game again.
- Input reading of Player's previously chosen settings, if any exist.
- Mouse to select settings.

Frequency of Occurences:

- When the Player uses the Settings menu.

Open Issues:

- How will the settings chosen affect the accessibility of the game?

Achieve Goal

Scope: Clickster
Primary Actor: Player
Stakeholders and Interests:

Player: Keep the player engaged and after reaching the end goal, the player wins the game.

Developer: Implements an end goal that the player must reach to win.

Precondition:

Players must surpass all milestones and reach the end goal to be able to win the game.

Success Guarantee:

Player achievements are saved throughout the game. Once the end goal is reached, the Player wins the game.

Main Success Scenario:

1. Player launches the game.
2. System loads the game.
3. User plays through the game.
4. System tracks the user.
5. Player makes progress in the game.
6. System tracks progress.
7. Player reaches milestones.
8. System keeps track of milestones.
9. Player reaches the end goal.
10. System tracks the end milestone.
11. Players are notified that they won the game.

Extensions:

*a. At any time, the user wants to play the game again after winning.

1. Player achieves the end goal.
2. Player wins the game.
3. Winning screen pops up.
4. Players can keep playing the game.

Special Requirements:

- Winning game screen pops up after the end goal is reached to let the player know that they won the game.
- Players should be informed of a certain end goal that they need to reach.

Technology and Data Variations List:

-Mouse to click through the game.

Frequency of Occurences:

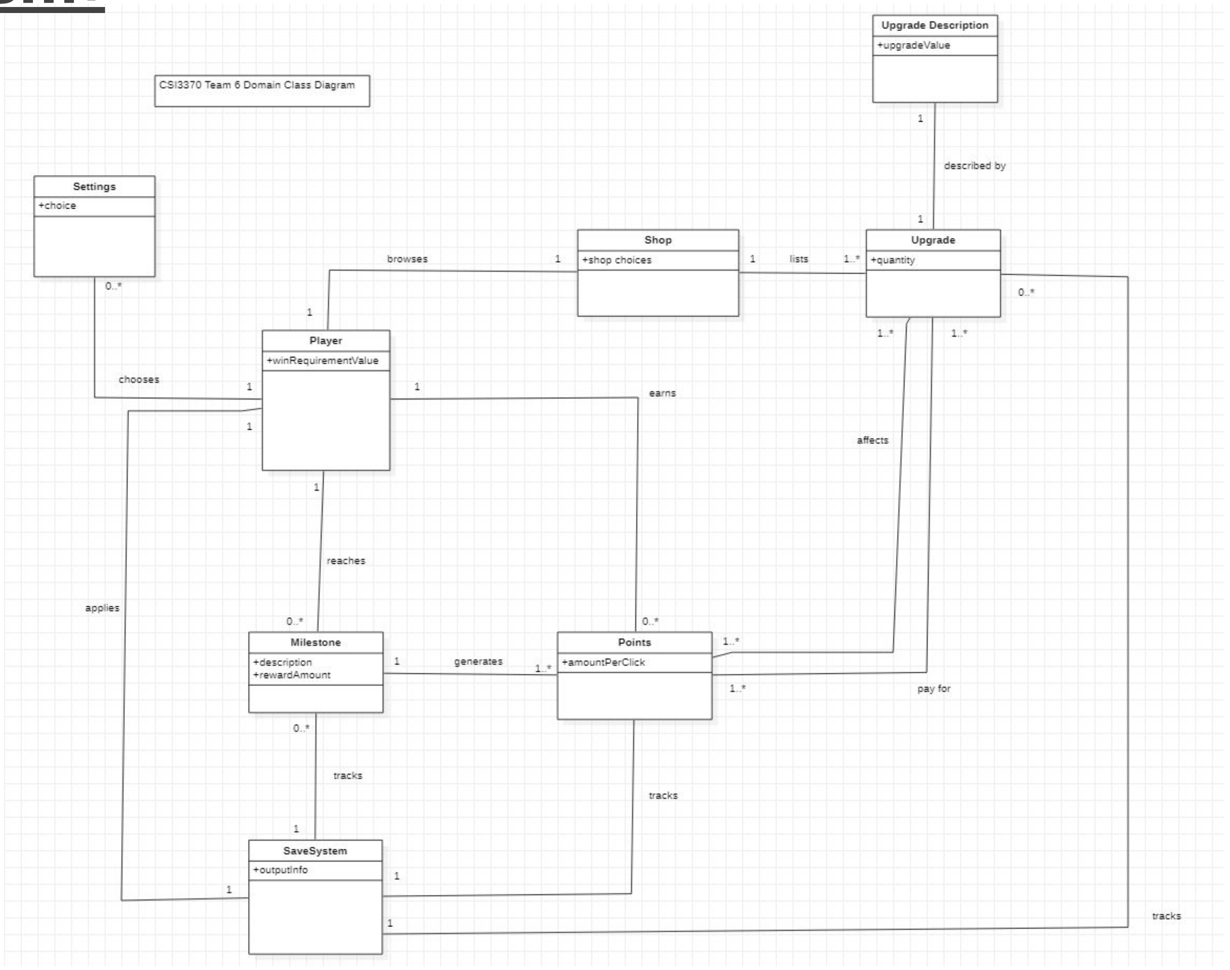
Winning screen pops up at the end.

Open Issues:

Can the user lose the game?

Domain Class Diagram:

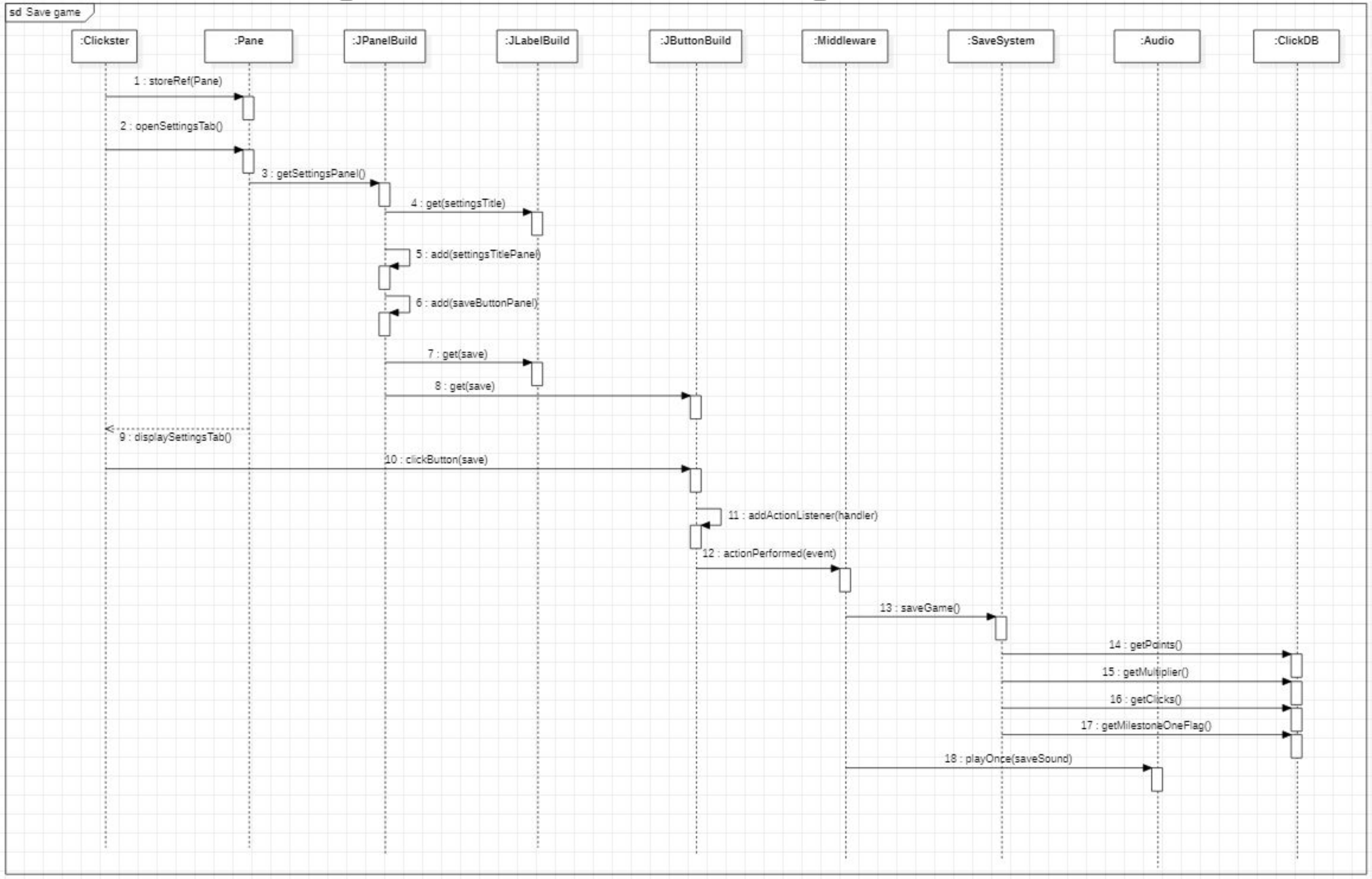
- The Player is the core of the whole application in the domain view.
- The Player can choose from settings, browse the shop with upgrades in it, save the game, reach milestones, and earn points.
- Most things are tracked by the SaveSystem, such as points, upgrades, and milestones. These are applied to the Player constantly.
- Upgrades have descriptions in the shop.
- Settings contain choices for the player to choose from.



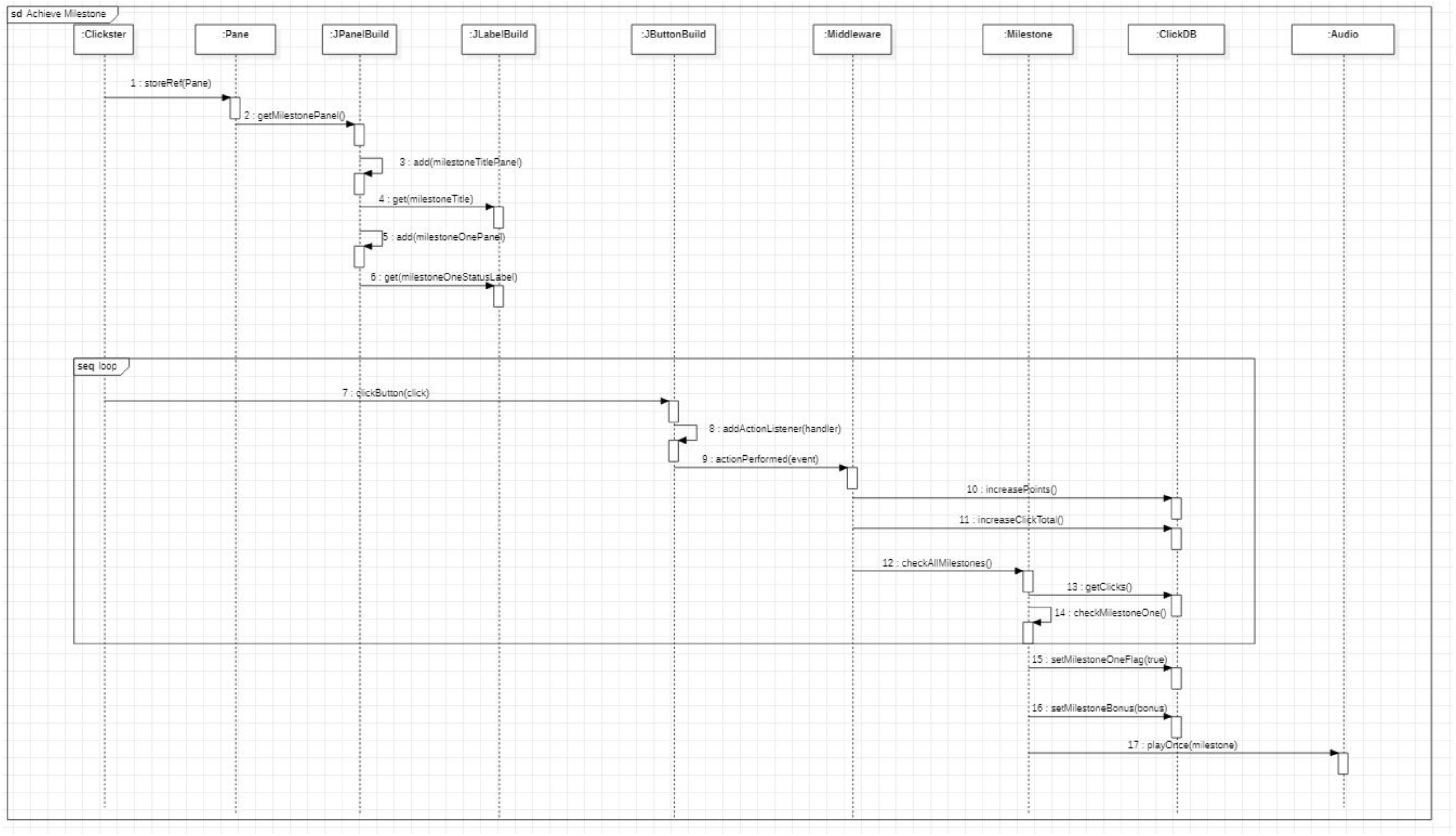
Phase 3

Design Modeling

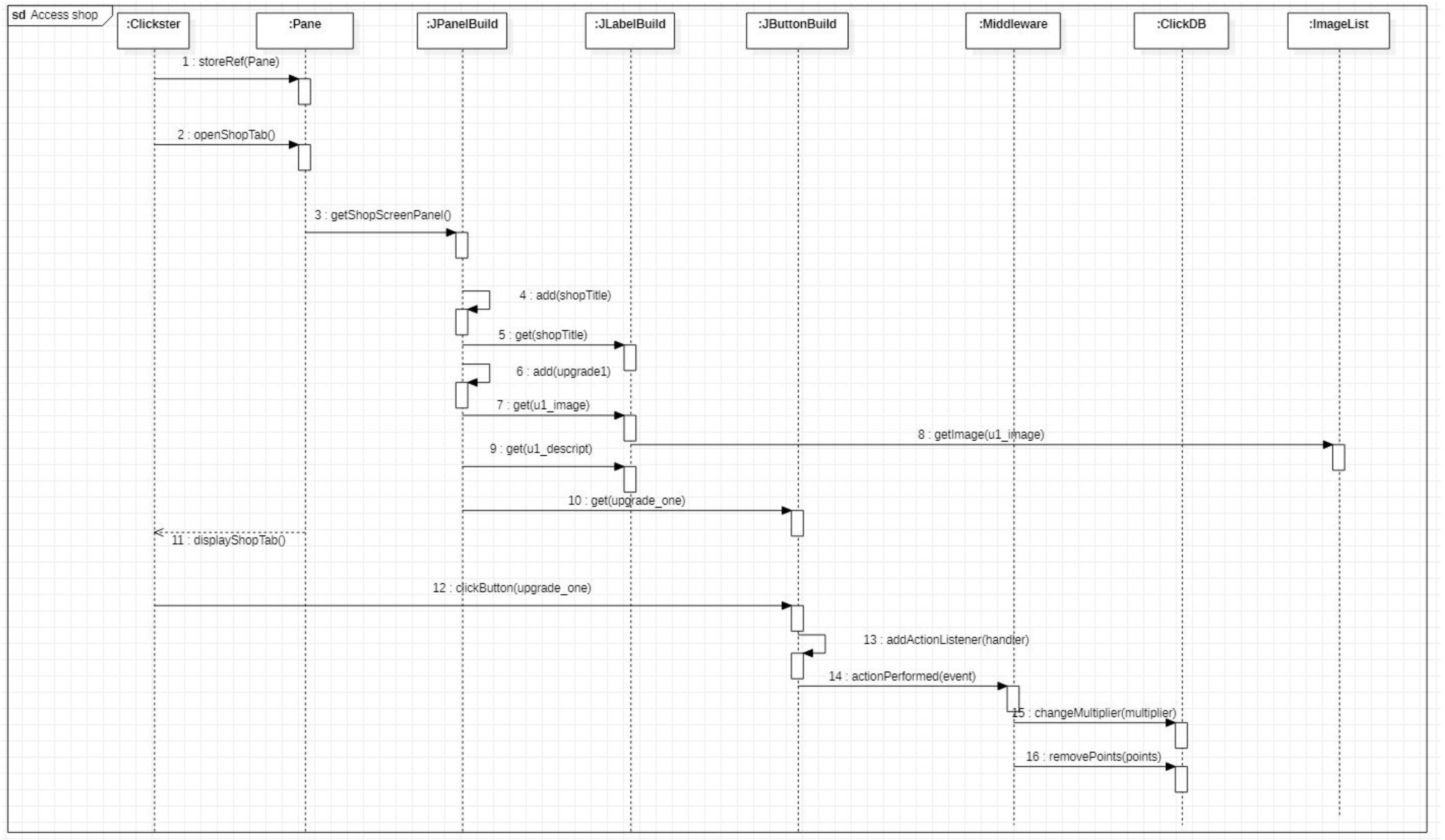
Save Game Design Sequence Diagram:



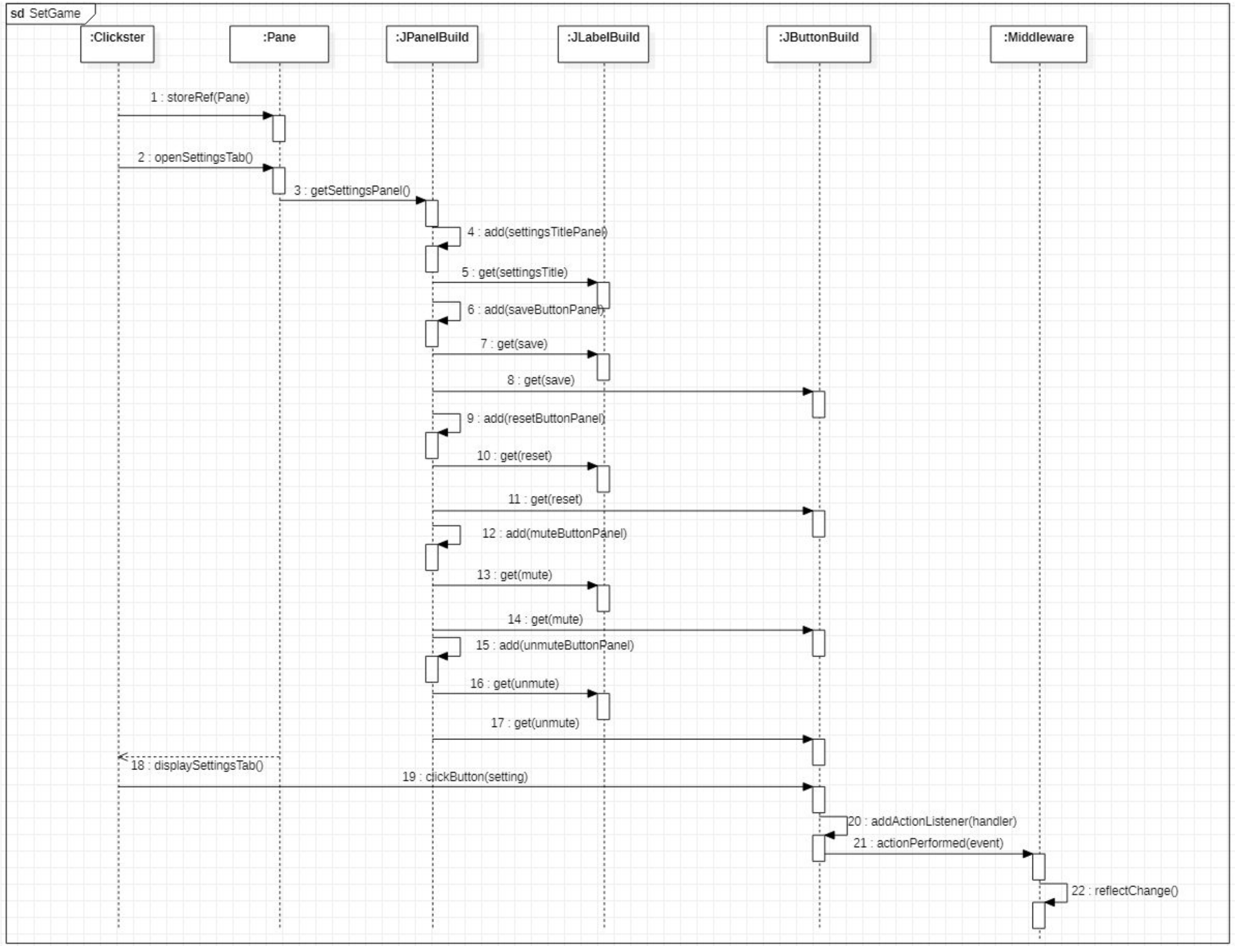
Achieve Milestone Design Sequence Diagram:



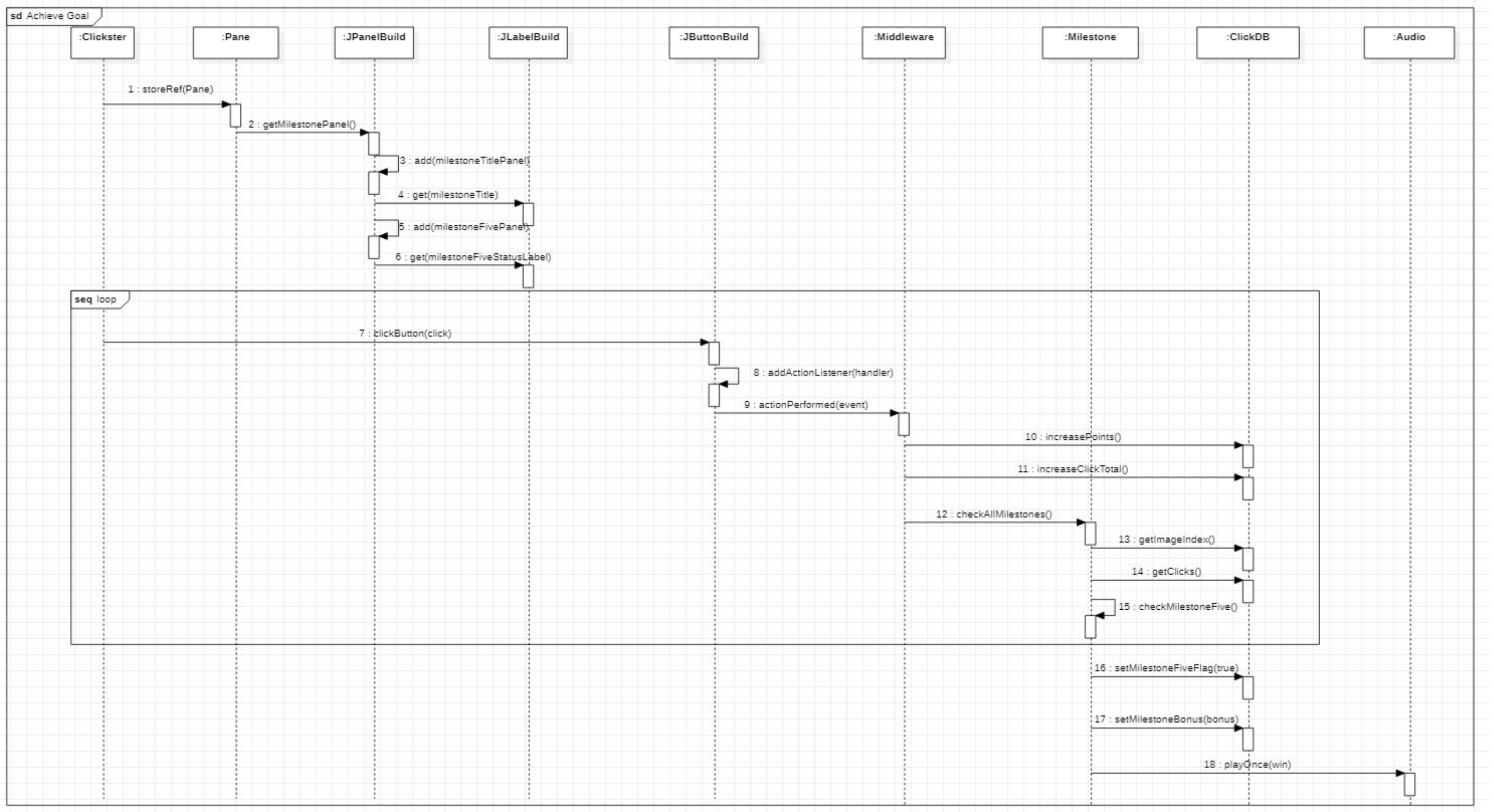
Access Shop Design Sequence Diagram:



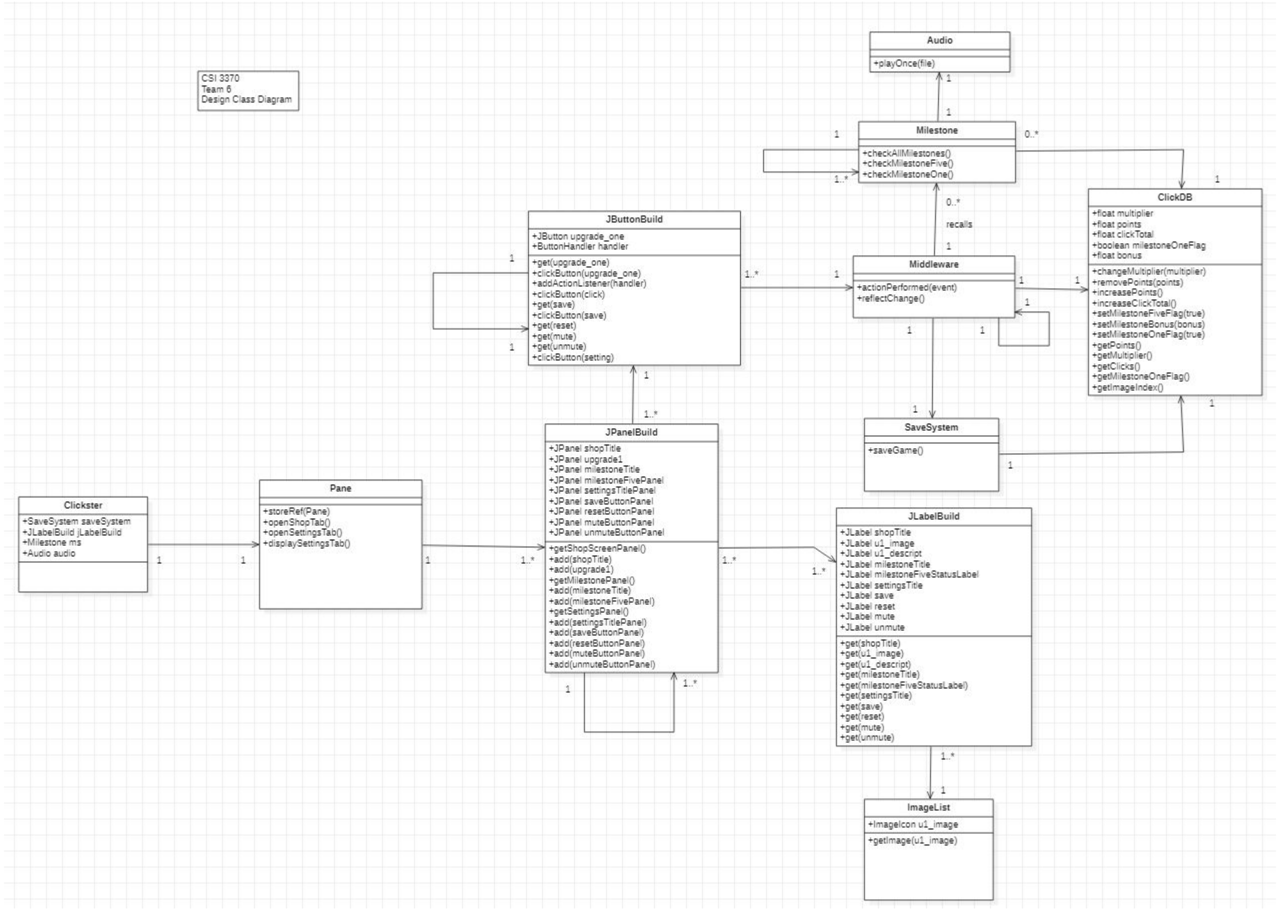
Set Game Design Sequence Diagram:



Achieve Goal Design Sequence Diagram:


sd Achieve Goal
:Clickster
:Pane
:JPanelBuild
:JLabelBuild
:JButtonBuild
:Middleware
:Milestone
:ClickDB
:Audio
1 : storeRef(Pane)
2 : getMilestonePanel()
3 : add(milestoneTitlePanel)
4 : get(milestoneTitle)
5 : add(milestoneFivePanel)
6 : get(milestoneFiveStatusLabel)
seq loop
7 : clickButton(click)
8 : addActionListener(handler)
9 : actionPerformed(event)
10 : increasePoints()
11 : increaseClickTotal()
12 : checkAllMilestones()
13 : getImageIndex()
14 : getClicks()
15 : checkMilestoneFive()
16 : setMilestoneFiveFlag(true)
17 : setMilestoneBonus(bonus)
18 : playOnce(win)

Design Class Diagram:



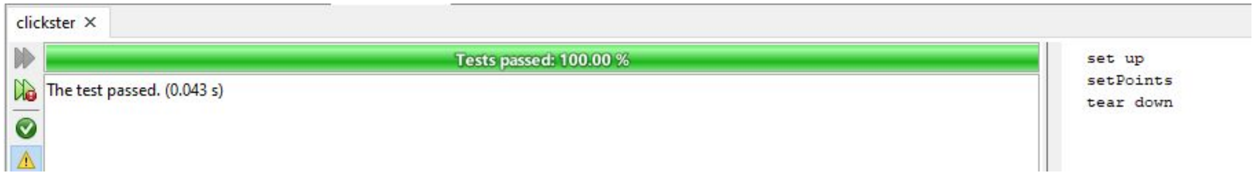
Phase 4

Testing

All JUnit 4 tests done for ClickDB.java, a database class.

```
    @Test
    public void testSetPoints() {
        System.out.println("setPoints");
        float expectedPoints = 25;
        clickDB.setPoints(points);
        assertTrue(expectedPoints == clickDB.getPoints());
    }

    @Test
    public void testSetMultiplier() {
        System.out.println("setMultiplier");
        float expectedMultiplier = 2.0f;
        clickDB.setMultiplier(multiplier);
        assertTrue(expectedMultiplier == clickDB.getMultiplier());
    }

    @Test
    public void testSetMilestoneFlagOne() {
        System.out.println("setMilestoneFlagOne");
        boolean expectedFlag = true;
        clickDB.setMilestoneOneFlag(milestoneOneFlag);
        assertTrue(expectedFlag ==
clickDB.getMilestoneOneFlag());
    }
}
```

testSetPoints() passing

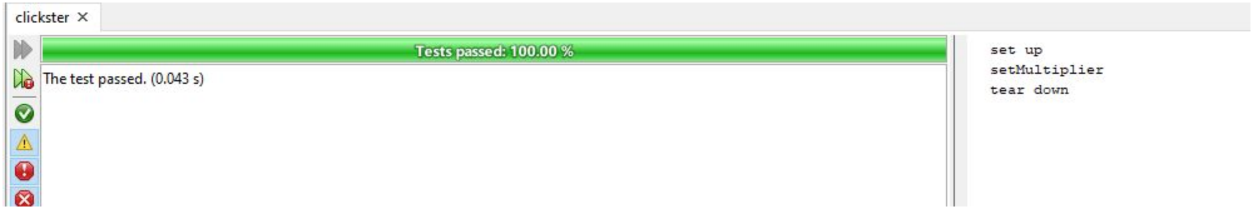


Tests passed: 100.00 %

The test passed. (0.043 s)

```
set up
setPoints
tear down
```

testSetMultiplier() passing

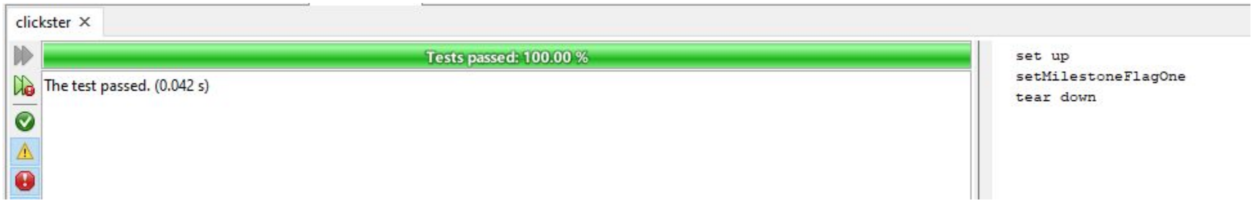


Tests passed: 100.00 %

The test passed. (0.043 s)

```
set up
setMultiplier
tear down
```

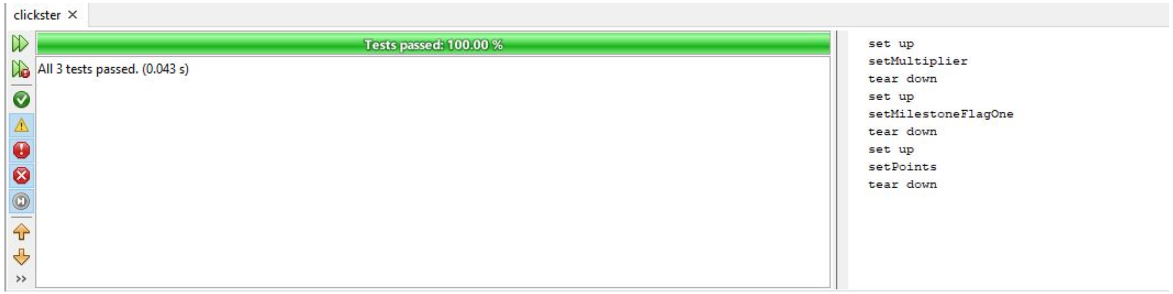
testSetMilestoneOneFlag() passing



```
Tests passed: 100.00 %
The test passed. (0.042 s)
```

```
set up
setMilestoneFlagOne
tear down
```

All tests passing at once



clickster ✕

Tests passed: 100.00 %

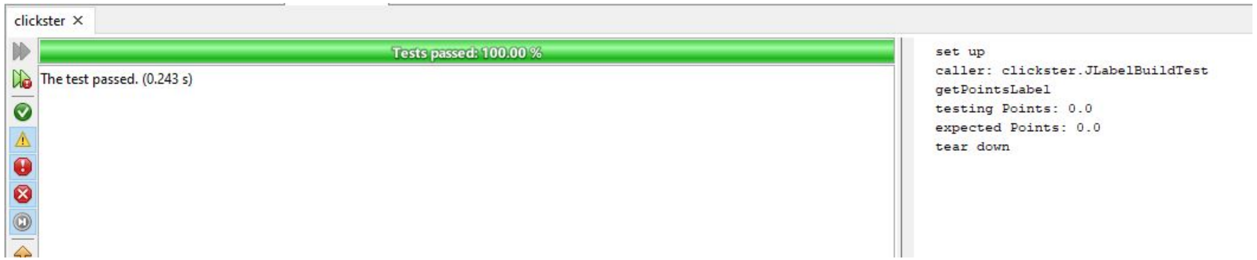
All 3 tests passed. (0.043 s)

```
set up
setMultiplier
tear down
set up
setMilestoneFlagOne
tear down
set up
setPoints
tear down
```

All JUnit 4 tests done for JLabelBuild.java, a user interface class for JLabels.

```
        @Test
        public void testGetPointsLabel() {
        System.out.println("getPointsLabel");
        JLabel expectedLabel = new JLabel("Points: " + 0.0);
        System.out.println("testing " + jLabelBuild.getPointsLabel().getText());
        System.out.println("expected " + expectedLabel.getText());
        assertTrue(jLabelBuild.getPointsLabel().getText().equals(expectedLabel.getText()));
        }

        @Test
        public void testGetClicksLabel() {
        System.out.println("getClicksLabel");
        JLabel expectedLabel = new JLabel("Clicks: " + 0.0);
        System.out.println("testing " + jLabelBuild.getClicksLabel().getText());
        System.out.println("expected " + expectedLabel.getText());
        assertTrue(jLabelBuild.getClicksLabel().getText().equals(expectedLabel.getText()));
        }

        @Test
        public void testGetMilestoneOneStatusLabel() {
        System.out.println("getMilestoneOneStatusLabel");
        JLabel expectedLabel = new JLabel("<html> <b> <center> Milestone One: Getting Started!
         </center> </b>Click 50 times for +0.5 points bonus! Your milestone status currently is: " +
         clickDB.getMilestoneOneFlag() + "</html> ");
        System.out.println("testing " + jLabelBuild.getMilestoneOneStatusLabel().getText());
        System.out.println("expected " + expectedLabel.getText());

        assertTrue(jLabelBuild.getMilestoneOneStatusLabel().getText().equals(expectedLabel.getText())
        );
        }

}
```

testGetPointsLabel() passing

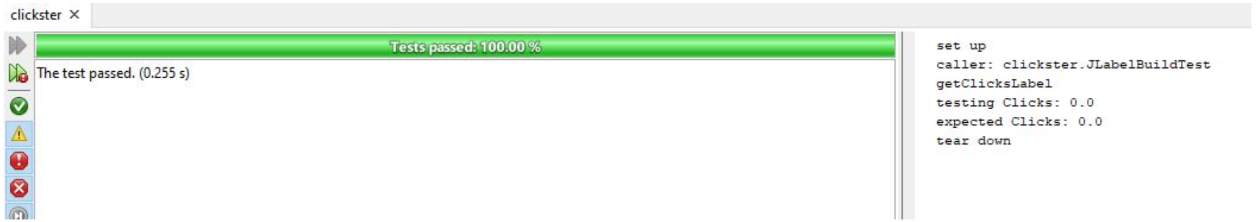


Tests passed: 100.00 %

The test passed. (0.243 s)

```
set up
caller: clickster.JLabelBuildTest
getPointsLabel
testing Points: 0.0
expected Points: 0.0
tear down
```

testGetClicksLabel() passing

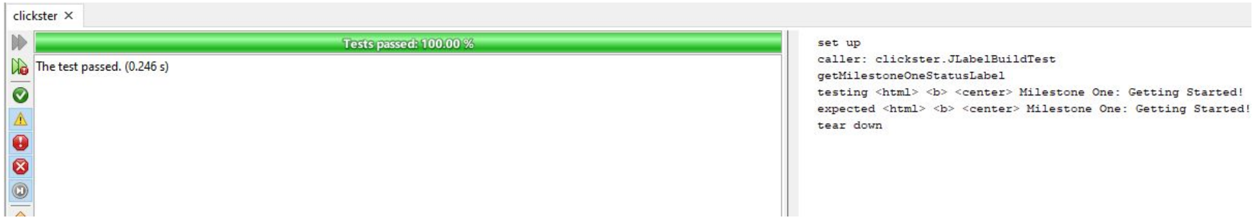


Tests passed: 100.00 %

The test passed. (0.255 s)

```
set up
caller: clickster.JLabelBuildTest
getClicksLabel
testing Clicks: 0.0
expected Clicks: 0.0
tear down
```

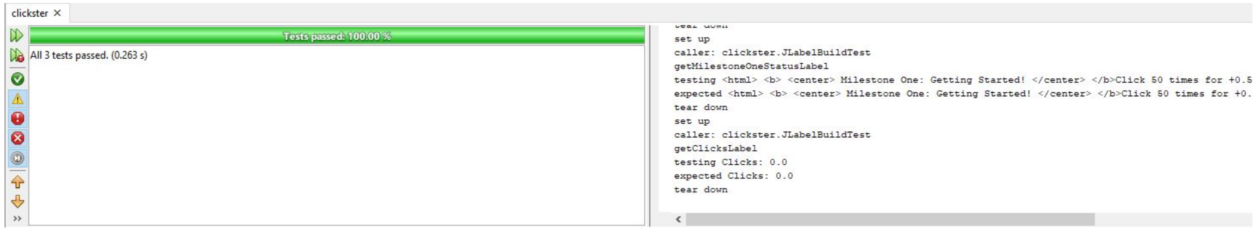
testGetMilestoneOneStatusLabel() passing



The test passed. (0.246 s)

```
set up
caller: clickster.JLabelBuildTest
getMilestoneOneStatusLabel
testing <html> <b> <center> Milestone One: Getting Started!
expected <html> <b> <center> Milestone One: Getting Started!
tear down
```

All tests passing at once



All 3 tests passed. (0.263 s)

```
tear down
set up
caller: clickster.JLabelBuildTest
getMilestoneOneStatusLabel
testing <html> <b> <center> Milestone One: Getting Started! </center> </b>Click 50 times for +0.5
expected <html> <b> <center> Milestone One: Getting Started! </center> </b>Click 50 times for +0.
tear down
set up
caller: clickster.JLabelBuildTest
getClicksLabel
testing Clicks: 0.0
expected Clicks: 0.0
tear down
```

Demo

References

References

Calanda, L. (2020, September 27). *#no copyright trumpet sound effect trumpet fanfare(welcome)*. Www.youtube.com; Liezel Calanda. https://www.youtube.com/watch?v=gxviN4BOVmw

Clip (Java Platform SE 7). (2020, June 24). Retrieved March 27, 2022, from https://docs.oracle.com/javase/7/docs/api/javax/sound/sampled/Clip.html

Heatley, B. (2015, October 31). *"8 Bit Journey!" Fun Adventure Chiptune Game Music by HeatleyBros.* Www.youtube.com; Heatley Music Publishing, ASCAP. https://www.youtube.com/watch?v=Vy8mVpTON3I

ImageIcon (Java Platform SE 7). (2020, June 24). Retrieved March 27, 2022, from https://docs.oracle.com/javase/7/docs/api/javax/swing/ImageIcon.html

References

Kim, D (2022, January 19). Metrics [PowerPoint slides]. Oakland University.

Most popular royalty free music | FiftySounds. FiftySounds. (n.d.). Retrieved March 27, 2022, from https://www.fiftysounds.com/royalty-free-music/

ReQTest. (2020, July 28). Functional vs non-functional requirements - understand the difference. ReQtest. Retrieved January 23, 2022, from https://reqtest.com/requirements-blog/functional-vs-non-functional-requirements/

Sketchpad (n.d.). Retrieved March 11, 2022 from https://sketch.io/

Swing - layouts. (n.d.). Retrieved March 27, 2022, from https://www.tutorialspoint.com/swing/swing_layouts.htm