● **Project Name: Escape the room**

● **Project Description:**

The product backlog: We are intended to design an escape game, which allow multiple players to login and join current games. The goal of the game is to find more hints to escape. There are many items in the room, such as beds, chests, windows, tables, chairs, paintings and other furniture.

1. Move: Players can press “↑” to move forward, “↓” to move backward, “←” to move left, and “→” to move right.
2. Trigger events: When a player walk near a item, he can press “F” to trigger events. He may pick up a key, a small piece of note, or face a keyboard to open a lock.
3. View a item: When a player owns a item in his inventory, he can view the description or status of the item.
4. Multiplayers: The online escape room could accept multiplayers in a room. They can cooperate to achieve the level.
5. Optional mini games: There may be some mini games for some triggered events or items. For example, we may write a sudoku on a paper note to hint the password for a lockbox in the room.

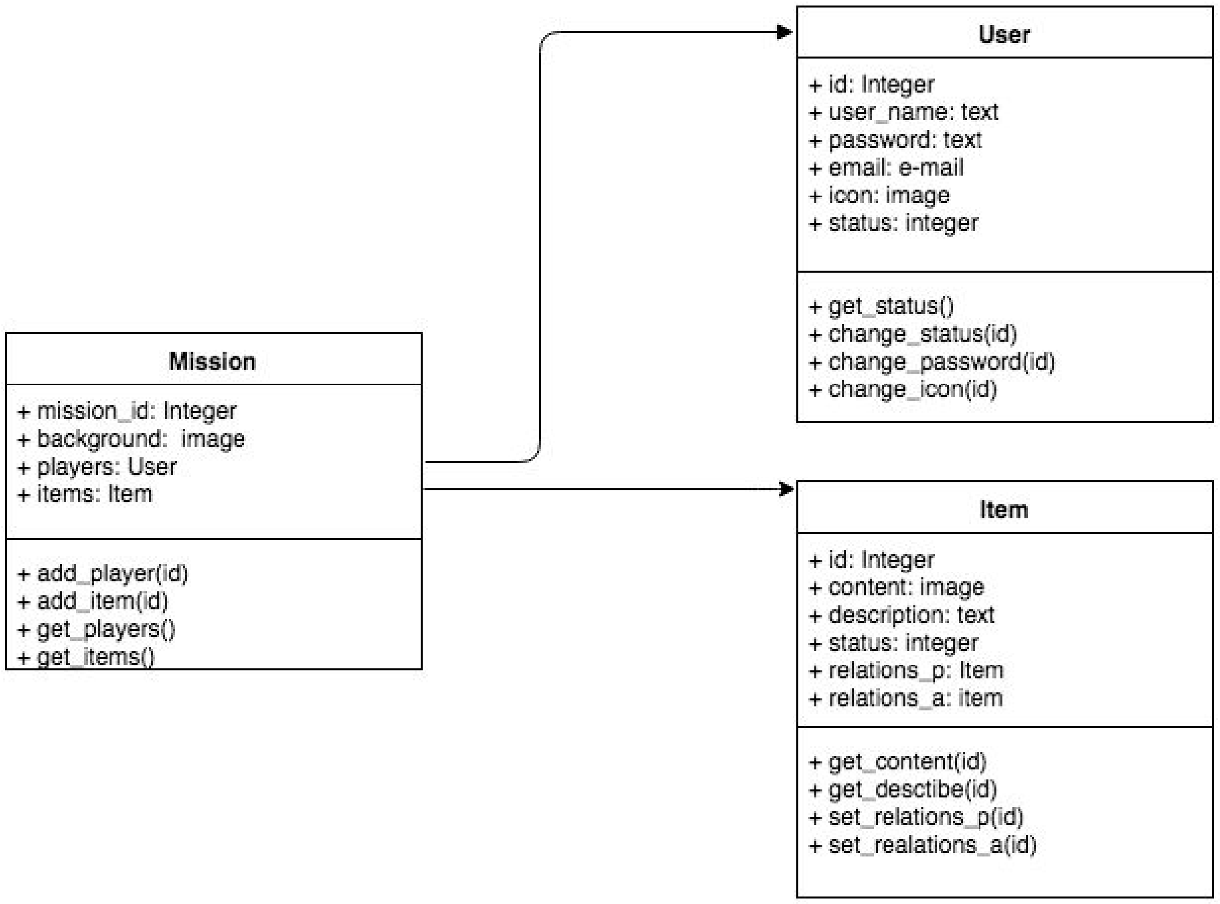
● The first sprint backlog: a complete list of the functionality you will complete during your first sprint, and how that work is allocated among your team members.

1. Front: Zhuoni Yang
    a. The basic user interface of the website (may be improved in the next sprint)
    b. Room model for the game
    c. Player model
    d. Send player’s location change to backend
2. End: Wenxiao Zhang
    a. Register
    b. Login
    c. Handle multiple players

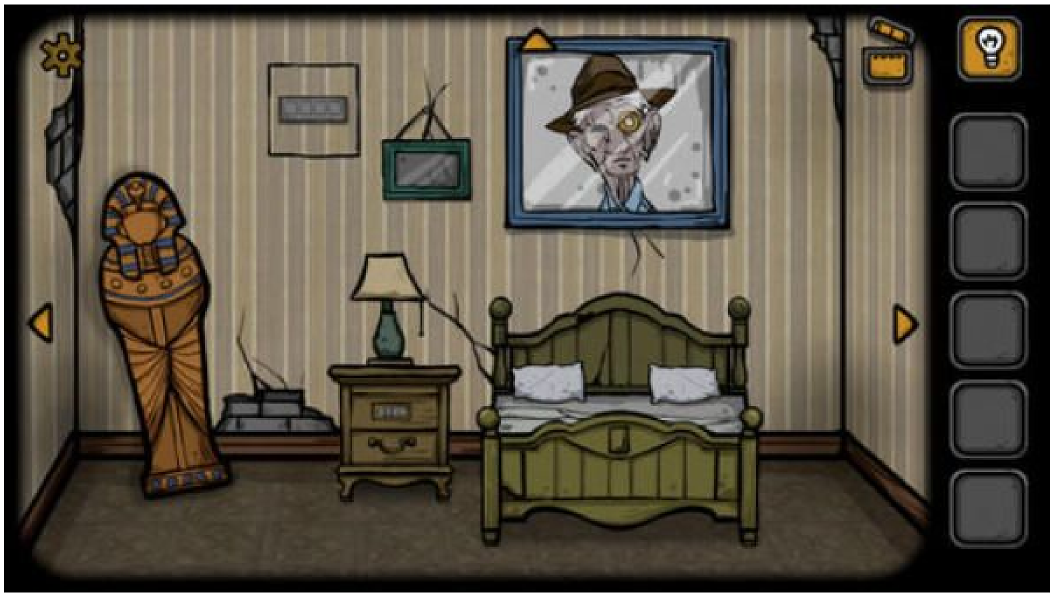
● The name and Andrew ID of the product owner for the first sprint.

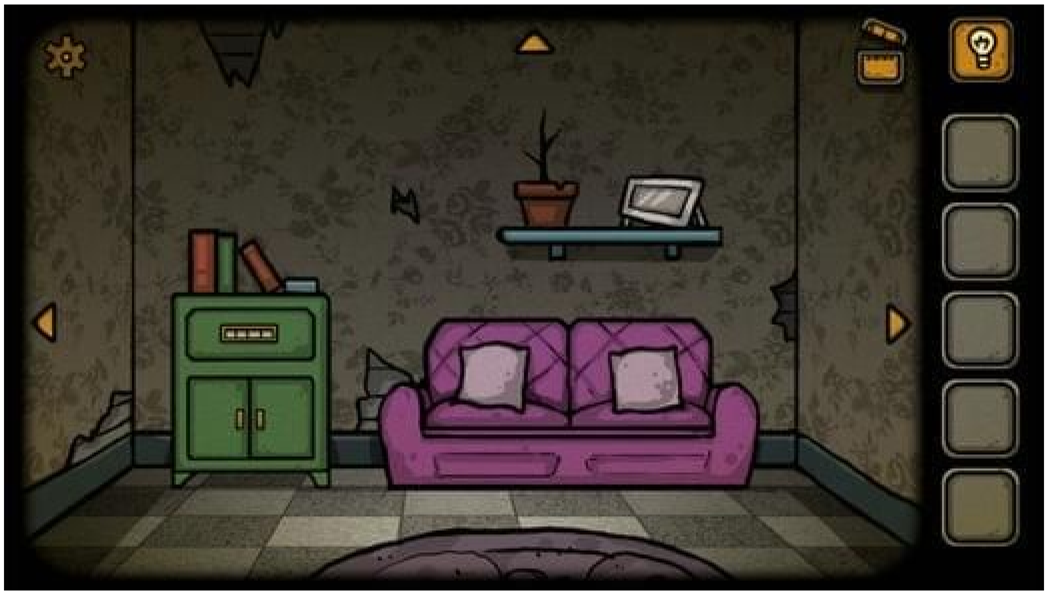
Zhuoni Yang (zhuoniy)
Wenxiao Zhang (wenxiaoz)

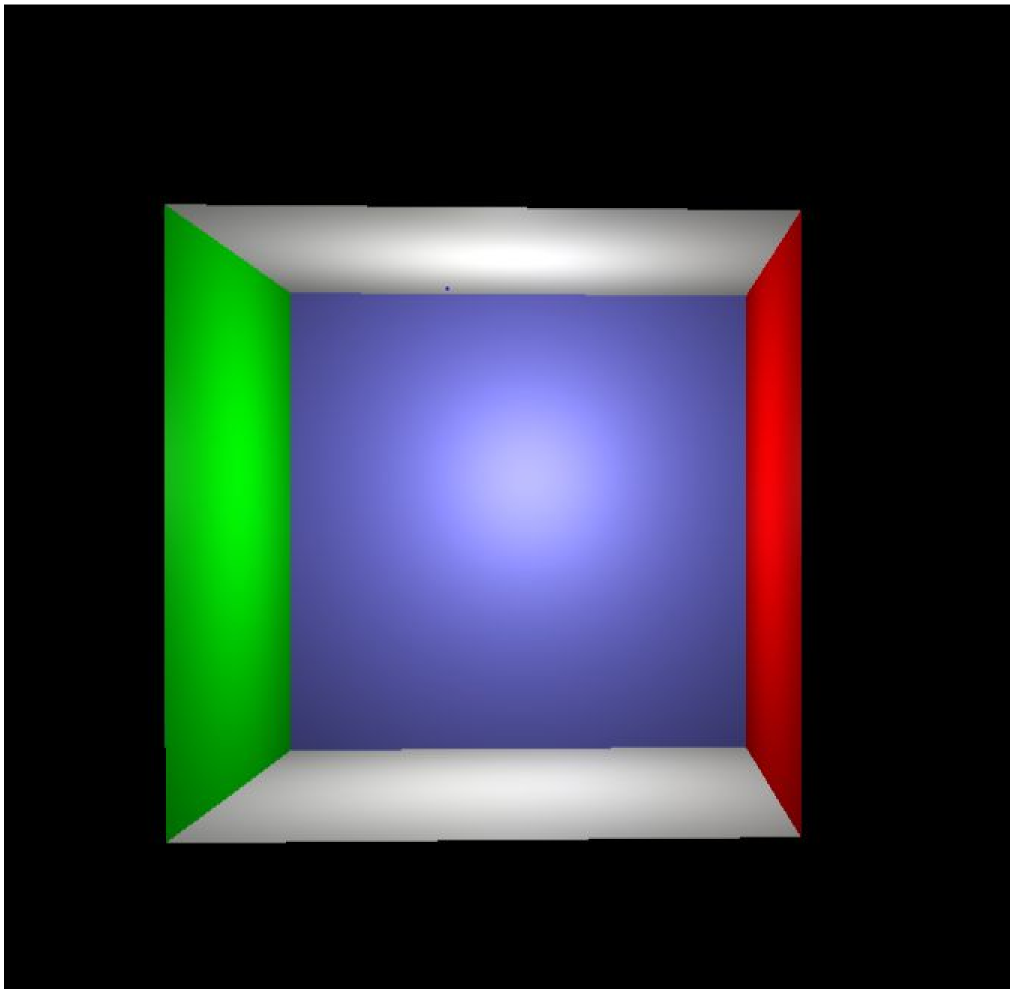
● A complete implementation of the data models used by your application. This may be written in as Django models, SQL, or some equivalent style implementation if you use another framework.



● A complete set of drawn wireframes or HTML mockups for your application, for all non-trivial views within the application.







Mockup of empty room

Source Code:

```
<!DOCTYPE html>
<html lang="en">
	<head>
		<title>three.js webgl - mirror</title>
		<meta charset="utf-8">
		<meta name="viewport" content="width=device-width, user-scalable=no,
minimum-scale=1.0, maximum-scale=1.0">
		<style>
			body {
				color: #888888;
				font-family:Monospace;
				font-size:13px;

				background-color: #000;
				margin: 0px;
				overflow: hidden;
			}

			#info {
				position: absolute;
				top: 0px;
				width: 200px;
				left: calc(50% - 100px);
				text-align: center;
			}

			a {
				color: #00f;
			}
		</style>
	</head>
	<body>

		<div id="container"></div>
		<div id="info"><a href="http://threejs.org" target="_blank"
rel="noopener">three.js</a> - mirror
		</div>

		<script src="../build/three.js"></script>
		<script src="js/objects/Reflector.js"></script>
		<script src="js/controls/OrbitControls.js"></script>
```

```
        <script>

                // scene size
                var WIDTH = window.innerWidth;
                var HEIGHT = window.innerHeight;

                // camera
                var VIEW_ANGLE = 45;
                var ASPECT = WIDTH / HEIGHT;
                var NEAR = 1;
                var FAR = 500;

                var camera, scene, renderer;

                var cameraControls;

                var sphereGroup, smallSphere;

                init();
                animate();

                function init() {

                        var container = document.getElementById( 'container' );

                        // renderer
                        renderer = new THREE.WebGLRenderer();
                        renderer.setPixelRatio( window.devicePixelRatio );
                        renderer.setSize( WIDTH, HEIGHT );
                        container.appendChild( renderer.domElement );

                        // scene
                        scene = new THREE.Scene();

                        // camera
                        camera = new THREE.PerspectiveCamera( VIEW_ANGLE,
ASPECT, NEAR, FAR );
                        camera.position.set( 0, 75, 160 );

                        cameraControls = new THREE.OrbitControls(camera,
renderer.domElement);
                        cameraControls.target.set( 0, 40, 0);
```

```
				cameraControls.maxDistance = 400;
				cameraControls.minDistance = 10;
				cameraControls.update();


				var planeGeo = new THREE.PlaneBufferGeometry( 100.1, 100.1
);

				// reflectors/mirrors

				var geometry = new THREE.CircleBufferGeometry( 40, 64 );
				var groundMirror = new THREE.Reflector( geometry, {
					clipBias: 0.003,
					textureWidth: WIDTH * window.devicePixelRatio,
					textureHeight: HEIGHT * window.devicePixelRatio,
					color: 0x777777,
					recursion: 1
				} );
				groundMirror.position.y = 0.5;
				groundMirror.rotateX( - Math.PI / 2 );
				// scene.add( groundMirror );

				var geometry = new THREE.PlaneBufferGeometry( 100, 100 );
				sphereGroup = new THREE.Object3D();
				scene.add( sphereGroup );

				var geometry = new THREE.CylinderGeometry( 0.1, 15 *
Math.cos( Math.PI / 180 * 30 ), 0.1, 24, 1 );
				var material = new THREE.MeshPhongMaterial( { color: 0xffffff,
emissive: 0x444444 } );
				var sphereCap = new THREE.Mesh( geometry, material );
				sphereCap.position.y = -15 * Math.sin( Math.PI / 180 * 30 ) - 0.05;
				sphereCap.rotateX(-Math.PI);

				var geometry = new THREE.SphereGeometry( 15, 24, 24, Math.PI
/ 2, Math.PI * 2, 0, Math.PI / 180 * 120 );

				var geometry = new THREE.IcosahedronGeometry( 5, 0 );
				var material = new THREE.MeshPhongMaterial( { color: 0xffffff,
emissive: 0x333333, flatShading: true } );

				// walls
```

```
                                var planeTop = new THREE.Mesh( planeGeo, new
THREE.MeshPhongMaterial( { color: 0xffffff } ) );
                                planeTop.position.y = 100;
                                planeTop.rotateX( Math.PI / 2 );
                                scene.add( planeTop );

                                var planeBottom = new THREE.Mesh( planeGeo, new
THREE.MeshPhongMaterial( { color: 0xffffff } ) );
                                planeBottom.rotateX( - Math.PI / 2 );
                                scene.add( planeBottom );

                                var planeFront = new THREE.Mesh( planeGeo, new
THREE.MeshPhongMaterial( { color: 0x7f7fff } ) );
                                planeFront.position.z = 50;
                                planeFront.position.y = 50;
                                planeFront.rotateY( Math.PI );
                                scene.add( planeFront );

                                var planeRight = new THREE.Mesh( planeGeo, new
THREE.MeshPhongMaterial( { color: 0x00ff00 } ) );
                                planeRight.position.x = 50;
                                planeRight.position.y = 50;
                                planeRight.rotateY( - Math.PI / 2 );
                                scene.add( planeRight );

                                var planeLeft = new THREE.Mesh( planeGeo, new
THREE.MeshPhongMaterial( { color: 0xff0000 } ) );
                                planeLeft.position.x = -50;
                                planeLeft.position.y = 50;
                                planeLeft.rotateY( Math.PI / 2 );
                                scene.add( planeLeft );

                                // lights
                                var mainLight = new THREE.PointLight( 0xcccccc, 1.5, 250 );
                                mainLight.position.y = 60;
                                scene.add( mainLight );

                                var greenLight = new THREE.PointLight( 0x00ff00, 0.25, 1000 );
                                greenLight.position.set( 550, 50, 0 );
                                scene.add( greenLight );

                                var redLight = new THREE.PointLight( 0xff0000, 0.25, 1000 );
                                redLight.position.set( - 550, 50, 0 );
```

```
				scene.add( redLight );

				var blueLight = new THREE.PointLight( 0x7f7fff, 0.25, 1000 );
				blueLight.position.set( 0, 50, 550 );
				scene.add( blueLight );

			}

			function animate() {

				requestAnimationFrame( animate );

				var timer = Date.now() * 0.01;

				sphereGroup.rotation.y -= 0.002;

				smallSphere.position.set(
					Math.cos( timer * 0.1 ) * 30,
					Math.abs( Math.cos( timer * 0.2 ) ) * 20 + 5,
					Math.sin( timer * 0.1 ) * 30
				);
				smallSphere.rotation.y = ( Math.PI / 2 ) - timer * 0.1;
				smallSphere.rotation.z = timer * 0.8;

				renderer.render(scene, camera);

			}

		</script>
	</body>
</html>
```