Discuss how you plan to implement the project				
Architecture				
Packages	Subpackages	Interface / Class	Class -> implements interface or extends class	อธิบาย
AST (collect all type of Nodes that use in AST)	State เป็น Subpackage ในส่วนที่เอาไว้เก็บ Node ที่มีความเกี่ยวข้องกับ State เพื่อให้ง่ายต่อการค้นหา	Node 1. static class StateNode extends Node -public StateNode nextState; -public StateNode evaluate(Game game) 2. static class ExprNode extends Node -public long eval(Game game) -public String toString() อันนี้เป็น class ที่อยู่นอก Subpackages	AssignmentNode -private final String identifier; -private final Node.ExprNode expression; -public String getIdentifier() -public Node.ExprNode getExpression()	ตัวสีแดง คือ Interface ตัวสีฟ้า คือ class ตัวสีเขียว คือ enum
			DoneNode	class พวกนี้ extends StateNode
			Relocate	
			MoveNode -private final Direction direction;	
			InvestNode -private final Node.ExprNode expr; -private final String actionRe	
			CollectNode -private final Node.ExprNode expr;	
			AttackNode -private final Node.ExprNode expr; -private final Direction direction;	
			IfNode -protected final Node.ExprNode condition; -protected Node.StateNode trueBranch; -protected Node.StateNode falseBranch;	
			WhileNode extends IfNode -private long count; -private StateNode getLastState(StateNode state);	class นี้ extends IfNode เพื่อลดการเขียน code ซ้ำ
	Expr เป็น Subpackage ในส่วนที่เอาไว้เก็บ Node ที่มีความเกี่ยวข้องกับ Expression เพื่อให้ง่ายต่อการค้นหา		BinaryOperatorNode -private final String operator; -private final Node.ExprNode left; -private final Node.ExprNode right;	class พวกนี้ extends ExprNode
			NuberNode -private long value;	
			Identifier -private final String idf;	
			OpponentNode	
			NearbyNode -private final Direction direction;	
	-	-	ASTError extends RuntimeException -เอาไว้เก็บ Exception ที่เจอใน Packages AST เพื่อให้ง่ายต่อการอ่านและทำความเข้าใจ	

<table>
<tr><td rowspan="2">Parser</td><td rowspan="2">-</td><td>Parser
-Node.StateNode parse();</td><td>Parser_im
-Tokenizer tkz;
-list<String> Command;
-list<String> SpecialVariables;
-Node.StateNode parsePlan()
-Node.StateNode parseStatement()
-Node.StateNode parseStatements()
-Node.StateNode parseStatements()
-Node.StateNode parseBlockStatement()
-Node.StateNode parseIfStatement()
-Node.StateNode parseWhileStatement()
-Node.StateNode parseCommand()
-Node.StateNode parseAssignmentStatement()
-Node.StateNode parseActionCommand()
-Node.StateNode parseMoveCommand()
-Direction parseDirection()
-Node.StateNode parseInvestCommand()
-Node.StateNode parseCollectCommand()
-Node.StateNode parseShootCommand()
-Node.ExprNode parseExpression()
-Node.ExprNode parseTerm()</td><td></td></tr>
<tr><td>-</td><td>ParserError extends RuntimeException
-เอาไว้เก็บ Exception ที่เจอใน Parser_im
เพื่อให้ง่ายต่อการอ่านและทำความเข้าใจ</td><td></td></tr>
<tr><td rowspan="2">Tokenizer</td><td rowspan="2">-</td><td>Tokenizer
1. boolean hasNextToken();
2. String peek();
3. boolean peek(String s);
4. String consume();
5. boolean consume(String s);</td><td>Tokennizer_im
-private final String src;
-private int pos;
-private String next;</td><td></td></tr>
<tr><td>-</td><td>TokenizerError extends RuntimeException
-เอาไว้เก็บ Exception ที่เจอใน Tokenizer_im
เพื่อให้ง่ายต่อการอ่านและทำความเข้าใจ</td><td></td></tr>
<tr><td colspan="5">Code design</td></tr>
<tr><td colspan="5">data structures ในการเปลี่ยนแปลงครั้งนี้ไม่ได้ใช้ แต่ถ้าคล้ายๆ กันก็น่าจะเป็น class StateNode ที่มี Node.StateNode nextStateNode คล้ายคลึงกับ linkedlist</td></tr>
<tr><td colspan="5">ที่เปลี่ยนแปลงไปมากพอสมควรเพราะว่าต้องการให้เก็บข้อมูลเอาไว้ที่ Game State ที่ได้คิดเอาไว้ว่าเป็นส่วนที่เกี่ยวกับเกมมากที่สุด ทำให้โครงสร้างการเก็บข้อมูลในส่วนของ Grammar เปลี่ยนไปมากพอสมควร</td></tr>
</table>

ความเปลี่ยนแปลงที่เกิดขึ้นกับ Grammar

อย่างแรกโครงสร้างข้อมูลที่ได้ออกแบบใน Grammar ver เก่า นั้นถูกเปลี่ยนแปลงจากเดิมมากพอสมควรซึ่งจะบอกเป็นข้อๆ ไปดังนี้

1. Node ตอนแรกที่เราได้ออกแบบเป็น Interface เนื่องจากมี method ที่เราสามารถให้พูดคุยกันได้ แต่ว่าจากที่ได้ดูใน Grammar ทำให้ทราบว่า State กับ Expr มันต่อเนื่องแต่ไม่มีส่วนที่ต้องพูดคุยสื่อสารอะไรกัน ทำให้ต้องลบ method ที่ได้ออกแบบเอาไว้ใน Node ทิ้งไป ซึ่งทำให้ Interface ที่ว่างเปล่านั้นไม่มีอะไรให้ทำและดูเปล่าประโยชน์เกินไป จึงเปลี่ยนมันให้กลายเป็น class Node และให้ Node ย่อยที่เคยออกไปนั้นเป็น subclass ที่อยู่ใน Node ไปเลย ซึ่ง Node ที่ออกแบบไว้ตอนนี้มีอยู่ 2 Node คือ StateNode กับ ExprNode ซึ่งทั้ง 2 Node นี้จะทำงานในส่วนของพวกมันตามชื่อเลย
2. เนื่องจากการเปลี่ยนแปลงที่เกิดขึ้นในข้อ 1 ทำให้ทุกอย่างที่เคยใช้ Interface Node จะแก้เป็นไปใช้งาน Node.StateNode ไม่ก็ Node.ExprNode ตามที่เห็นใน Grammar ver ใหม่ ที่ส่งไปนะครับ
3. การเก็บข้อมูลของ ExprNode ในส่วนของ long eval(Map<String, Long> mem); ได้มีการเปลี่ยนแปลงภายในวงเล็บให้เป็น long eval(Game game); เนื่องจากมีความคิดใหม่ว่า การเก็บข้อมูลนั้นควรเอาไว้ในที่เดียวกัน และการที่ต้องทำ Game state ซึ่งเป็นส่วนที่น่าจะต้องเก็บข้อมูลเกือบทุกอย่างของเกมเอาไว้ทำให้เปลี่ยนที่เก็บข้อมูลจาก menไปเก็บข้อมูลใน game อย่างที่เห็นแทน ซึ่งในนั้นจะมี Map ของตัวมันอยู่
4. ซึ่งในส่วนของ StateNode ที่มี Node.StateNode evaluate(Game game) นั้น จะมีในส่วนของ AssignmentNode ที่ต้องเพิ่ม Node.StateNode evaluate(Map<String, Long> mem) เข้าไปด้วย เนื่องจากการเข้าถึงของทางฝั่งผู้เล่นที่จำเป็นต้องเก็บตัวแปรของตัวเองอย่างที่จารย์ชินเคยบอกเอาไว้ว่า ต่อให้ผู้เล่นทั้ง 2 ต่างก็ใช้ x เหมือนกัน แต่ก็ต้องไม่ไปปะปนกัน ดังนั้นในส่วนของ AssignmentNode จึงต้องเพิ่มเข้าไปอย่างที่บอกไปข้างต้นเพื่อให้เอา memory ของตัวเองเข้าไปบรรจุค่าตัวแปรที่เรา assign เข้าไป
5. ได้มีการสร้าง enum ที่ชื่อว่า Direction ขึ้นมาเพื่อเก็บทิศทางทั้ง 6 ของตัวเกมที่ต้องการเอาไว้ เพื่อให้เอาไปใช้กับ parseDirection ที่จะทำการอ่านทิศทางที่เราใส่ลงไปใน Construction plan และเปลี่ยนเป็นค่าที่เราสร้างเอาไว้ใน Direction ส่วนที่ว่าทำไมถึงสร้างขึ้นมานั้น เพื่อป้องกันความผิดพลาดที่อาจเกิดขึ้นระหว่างการเขียนโค้ด และทำให้เกิด HumanError แบบไม่ทันตั้งตัวได้
6. ได้มีการสร้าง List<String> Command และList<String> SpecialVariables ขึ้นมาใน Parser_im เพื่อให้ง่ายต่อการใช้งานและทำให้ตัว code ไม่ดูเปรอะจนเกินไป

7. จากในข้อ 6 List<String> SpecialVariables นั้นจะประกอบไปด้วยพวก command ต่างๆ อย่าง shoot, nearby, oppent และอื่นๆ ที่ถ้าผู้เล่นใช้งานคำพวกนี้เป็นตัวแปรจะแจ้งเตือนว่าห้ามใช้งาน และต้องกลับไปแก้ไข (ปล. สิ่งที่จารย์เขียนเอาไว้ใน ProjectSpec ก็ได้ใส่เอาไว้ในนี้แล้วด้วย)
8. แนวความคิดเกี่ยวกับ parseBlockStatement ตอนแรกคิดว่า BlockStatement นั้นจะเป็นในส่วนของตัวมันไปเลยอันอื่นจะไม่มีความเกี่ยวข้องกับ Block แต่หลังจากที่อ่านใน Construction plan อีกครั้ง ทำให้รู้ว่า Block นั้นสามารถใช้กับอะไรก็ได้ เหมือนกับการเขียนโค้ดปกติ ทำให้ BlockNode ที่ออกแบบเอาไว้ไม่ใช้งานอีกจึงหายไปจากระบบ แต่จำเป็นต้องเพิ่ม parse เข้ามาอีกอันหนึ่งชื่อว่า parseStatements เป็น parse ที่เอาไว้อ่านว่า State ต่อไปมีอยู่หรือไม่ ซึ่งมันจะวนอ่านไปจนกว่าจะไม่มี หรือว่าจะจบ BlockStatement
9. RegionNode ได้เปลี่ยนเป็น InvestNode กับ CollectNode แทน เพื่อให้ State ที่ได้นั้นต้องเป็นสิ่งที่เล็กที่สุดเท่าที่เป็นไปได้ตามหลักของ AST
10. ได้มีการสร้าง Error ขึ้นมาในแต่ละ Package ของ Grammar เพื่อให้สามารถอ่านได้อย่างสะดวกมากยิ่งขึ้นถ้าเราต้องการรู้ว่ามันจะเกิด Throw อะไรบ้างถ้าเราใช้สิ่งที่อยู่ใน Package นั้นๆ
11. ในส่วนของ Tokenizer ได้เพิ่มความสามารถที่อ่าน comment อย่างในตัวอย่างของ Construction plan ได้ และตัวแปรที่อีกฝ่ายเขียนมานั้นสามารถใส่ตัว _ ได้ เนื่องด้วยเหตุผลที่ว่าบางครั้งการเขียนติดกันอาจจะทำให้อ่านยากได้จึงคิดว่าการเพิ่มตัว _ นี้เข้าไปอาจจะทำให้อ่านตัวแปรแล้วรู้สึกโอเคมากขึ้น
12. ในส่วนของวิธีการอ่าน Construction plan นั้นได้เปลี่ยนไปใช้ File.readString() แทนการใช้ BufferedReader ที่เคยใช้มาเนื่องจาก readString นั้นจะเก็บข้อมูลที่เราอ่านได้จาก File มาเป็น String ตัวเดียว ส่วนถ้า File ที่มีหลายบรรทัดนั้น readString จะเก็บทุกอย่างอยู่ในบรรทัดเดียวแต่จุดที่ใน File ได้ขึ้นบรรทัดใหม่นั้นจะมีตัว \n ขั้นเอาไว้ ทำให้เวลาทดสอบ System.out.println() ออกมาจึงยังมีผลลัพธ์ไม่แตกต่างจาก File ที่เราไปอ่านมา ทำให้ในส่วนของ Tokenizer ได้มีการเพิ่มว่า ถ้าเจอ # หลังจากนั้นจะเป็นเนื้อหาของ comment จนกว่าจะเจอ \n ที่แปลว่าได้ขึ้นบรรทัดใหม่เป็นที่เรียบร้อยแล้ว
13. ซึ่งจากแนวคิดของข้อ 8 และข้อ 12 ที่ปูทางมาถึงขนาดนี้ทำให้ได้มีการเพิ่มบางอย่างเข้าไปใน StateNode ซึ่งก็คือ Node.StateNode nextState ถ้าอธิบายแบบง่ายๆ มันคือการที่ทุก Node จะมีการเชื่อมต่อไปยัง Node ถัดไป เพื่อให้ง่ายต่อการนำไปใช้งาน เพราะถ้า nextState เป็น null ก็จะสรุปได้ว่าการ parse นั้นได้จบลงไปแล้ว

14. จากข้อที่ 9 การเปลี่ยนแปลงที่เกิดขึ้นนั้นไม่ได้ส่งผลต่อโครงสร้างในส่วนของ Parser_im มีเพียงในส่วนที่ว่า InvestNode กับ CollectNode นั้นจะสนใจแต่ ExprNode เท่านั้น
15. จากการเปลี่ยนแปลงโครงสร้างหลักของ Node ในข้อ 1 และแก้ไขแนวคิดที่ออกแบบไว้คร่าวๆ ในครั้งแรก ทำให้การ evaluate ของ StateNode ต้องมีการ return ค่าที่เป็น StateNode เสมอ แทนที่ที่เคยเป็น void มาก่อน เพื่อให้นำไปใช้งานต่อได้ (การออกแบบในตอนแรกยังไม่ได้คิดถึงเรื่องการนำไปใช้งานต่อ)
16. มีการเปลี่ยนแปลงในส่วนของ WhileNode จากที่ว่าไป extends Node.StateNode เปลี่ยนไป extends IfNode แทน เนื่องจากโครงสร้างของ Node ทั้งสองนั้นมีความคล้ายคลึงที่มากพอสมควร จึงทำการ extends เพื่อนำ code มาใช้ซ้ำอย่างที่เคยเรียนมา ทำให้ในส่วนของ IfNode นั้นต้องเปลี่ยนจาก private เป็น protected ทั้งสามตัวอย่างที่เห็นใน Grammar ver ใหม่

What did your group learn from designing and implementing the parser and evaluator?

สิ่งที่ได้เรียนรู้จากการเขียน Grammar นั้นบอกเป็นข้อได้ดังต่อไปนี้

1. การออกแบบ เพราะสิ่งที่เราเคยออกแบบเอาไว้นั้นอาจจะไม่สามารถนำมาใช้ได้จริงทั้งหมดเพราะอาจจะยังไม่ได้คิดในเรื่องของการนำไปใช้งานต่อหรือลืมคิดเรื่อง บางอย่างไป ทำให้เวลาเขียนจริงมีการเปลี่ยนแปลงอย่างเลี่ยงไม่ได้
2. ได้ความรู้เรื่องใหม่ๆ มากพอสมควร อย่างการอ่านข้อมูลจาก File นั้นตอนแรกคิดจะใช้ BufferedReader อย่างที่เคยใช้ไปใน lab 09 ทว่าหลังจากที่ได้ไปลองถามจาก chatGPT และลองค้นหาข้อมูลใน Internet ทำให้ได้รู้ถึงทำสั่งอย่าง readString() ที่ทำให้เราไม่ต้องสนใจเรื่องการขึ้นบรรทัดใหม่ และต้องมาคิดถึงเรื่องของการอ่านบรรทัดใหม่อีก แค่ต้องไปเพิ่มให้ Tokrnizer_im ต้องอ่าน \n แล้วรู้ว่ามันคือการขึ้นรรทัดใหม่ จะแปลได้ว่าถ้าบรรทัดก่อนได้มีการเขียน comment เราจะเริ่มการเขียน command ใหม่ ไม่ใช่ comment นะ
3. การใช้งาน enum เป็นสิ่งที่ดีเพราะมันจะทำให้เราลดโอกาสเกิด HumanError ที่อาจเกิดจากการเขียน String ที่ผิดพลาดขึ้นมาได้ และทำให้เวลากลับมาอ่านโค้ดหรือทำการแก้ไขไม่ยุ่งยากจนเกินไป

Discuss how you plan to implement the project

Architecture

Root Component	Main Component	style	Sub Components	styles	Description
index.jsx	App.jsx	App.css	Landing.jsx	inputStyle.css	Home page
			InputName.jsx	landingStyle.css	กรอกชื่อผู้เล่น
			HexGrid.jsx	Gridstyle.css	วาดregionตามจำนวน กว้าง, ยาว
			Hexagon.jsx		วาดregion 1ช่อง

Code design

ออกแบบโดยแบ่ง componentใหญ่ๆออกเป็นชิ้นเล็กเพื่อความยืดหยุ่นในการออกแบบ และการแก้ไข แต่โครงสร้างของการสร้างmapนั้นยังมีจุดบอดคือจะต้องสร้างrowก่อนแล้วค่อยเอา row มาต่อกันอีกที ทำให้แต่ละ row จะเหมือนกัน ทำให้ยังไม่สามารถใช้งานเต็มร้อยได้

Testing: describe your testing plan

ทดสอบโดยการทำ user interact กับการ visualize เพื่อดูว่าพวกbuttonทำงานไหมแล้ว redirectเราไปถูกเพจรึเปล่า

Glassblock: เท่าที่ออกแบบเอาไว้คือจะหา testcase ที่จะทำการทดสอบว่า code ที่เราออกแบบไว้นั้นได้ใช้ในส่วนเราได้เขียนเอาไว้ตามกรณีที่เราคิดเอาไว้หรือไม่
อย่างถ้าเป็น tokenizer จะเห็นได้ง่ายที่สุดก็คงเป็นอย่าง method computeNext() ดูว่าถ้าเราใส่ testcase ที่แตกต่างกันมันจะเข้าไปในกรณีในแบบที่คิดเอาไว้หรือไม่

Discuss how you plan to implement the project

Architecture

Packages	Subpackages	Interface / Class	Class -> implements interface or extends class	อธิบาย
Game	-	Game -Map<String, Long> getIdentifiers(); -void attack(Direction direction, long money); -void collect(long money); -void invest(long money); -void move(Direction direction); -void relocate(); -long nearby(Direction direction); long opponent();	Game_im -private final Player player1; -private final Player player2; -private final List<Region> territory; -private final Map<String, Long> identifiers;	ตัวสีแดง คือ Interface ตัวสีฟ้า คือ class ตัวสีเขียว คือ enum
			ReadData -private static int row; -private static int col; -private static int init_plan_min; -private static int init_plan_sec; -private static double init_budget; -private static double init_center_dep; -private static int plan_rev_sec; -private static int plan_rev_min; -private static int rev_cose; -private static double max_dep; -private staticdouble interest_pct; -private static List<Region> territory; -public static void getDataFile(); -public static List<Region> createMap(); -public static Player createPlayer(String name);	class นี้ไม่ได้ extends อะไรทั้งสิ้น
Player	-	Player -String getName(); -double getBudget(); -void updateBudget(double money); -void moveCityCrew(String direction); -int getCityCrewLocation(); -int getCityCenterLocation(); -boolean life(); -long attack(String direction); -Map<String, Long> getIdentifiers();	Player_im -private final String name; -private double budget; -private boolean life; -private final int CityCrew; private final int CityCenter; -private final Map<String, Long> identifier;	
Region	-	Region -String getOwner(); -void updateOwner(String owner); -double get Budget(); -void updateBudget(double money); -int getLocation(); -int Oppent(); -int nearby();	Regionr_im -private final int location; -private String owner; -private double budget;	
-	-	-	GSError extends RuntimeException -เอาไว้เก็บ Exception ที่เจอใน Game_state ทั้งหมด เพื่อให้ง่ายต่อการอ่านและทำความเข้าใจ	

Code design

data structures ที่ผมคิดว่าจะใช้ในงานนี้ น่าจะเป็น List<Region> กับ Map<String, Long>

จากเท่าที่คิดและทดลองเขียนมา ข้อดี คือ ทำให้การเก็บข้อมูลของเกม UPBEAT นั้นจะในส่วนของ Game State อย่างเดียว ซึ่งจะทำให้ข้อมูลในส่วนอื่นของตัวเกมมีเพียงข้อมูลที่จำเป็นต้องใช้งานอย่างจริงจังเท่านั้น ข้อเสีย คือ ถ้าเราเก็บข้อมูลในส่วนนี้ไม่เป็นระเบียบอาจจะทำให้การนำไปใช้งานแต่ละทียุ่งยาก และต้องเสียเวลาไม่มากก็น้อย ดังนั้นการกำหนดชื่อของสิ่งที่เก็บให้เข้าใจง่ายเป็นสิ่งที่ค่อนข้างสำคัญในระดับหนึ่ง

Tools

1. Disscord ใช้ในการติดต่อสื่อสารและแจ้งเตือนเกี่ยวกับ project
2. Github ใช้ในการอัพเดท code ของ project และเพื่อให้ผู้ตรวจงานสามารถเข้ามาดูองค์ประกอบภายในของ project ได้
3. Google sheet ใช้ในการเขียนเกี่ยวกับ project โดยที่คนในกลุ่มสามารถเข้ามาแก้ไขหรือปรับแก้บางส่วนของรายงานที่จะส่งได้

จากการออกแบบข้างต้น ทางสมาชิกนกลุ่มได้หารือกันว่าโครงสร้างของตัว Game State ในตอนนี้ได้ตรงตามความต้องการที่ตัวเกมต้องการทุกอย่าง ทำให้ตัวเกมสามารถอ่านข้อมูลและส่งข้อมูลได้ตามที่ต้องการ เรียบง่ายและมีประสิทธิภาพมากที่สุด เพื่อลดภาระการประมวลผลที่จะมีระหว่างการเล่นเกมในมีเท่าที่จำเป็น
Testing: describe your testing plan
Blackblock: คิดว่าน่าจะทำการ test กับตัว Game เพียงอย่างเดียวเพราะ Game นั้นจะเป็นคนดึงการใช้งานในส่วนของ Player กับ Region ซึ่งการ test นั้นน่าจะแบ่งออกเป็น การทดสอบใส่ข้อมูลว่าการรับข้อมูลจากทาง frontend ด้วย file สกุล json นั้นเป็นไปตามที่ควรจะเป็นหรือไม่ ถ้าการที่เราใส่ข้อมูลไปไม่ครบนั้นจะทำให้เกิดปัญหาอย่างที่คิดเอาไว้หรือไม่, การทดสอบการสร้างแผนที่ของเกมขึ้นมาว่าการที่เราใส่ row กับ col เข้าไปนั้น territory จะต้องมีจำนวน region ตรงตามที่ควรจะเป็นหรือไม่, การทดสอบสุดท้ายจะเป็นในส่วนของการทดสอบสร้างตัวละครของผู้เล่นขึ้นมา ซึ่งจะทดลองว่าถ้าเราใส่ชื่อเข้าไปแบบไหนจะทำให้เกิดผลลัพธ์เป็นอย่างที่ออกแบบเอาไว้หรือไม่ อย่างถ้าเราไม่ได้สร้าง map แล้วไปสร้างตัวละครเลย ก็ควรที่จะไม่ผ่าน หรือไม่ก็ถ้า row หรือ col เป็น 0 ก็ไม่ควรที่จะสร้างตัวละครขึ้นมาได้
Glassblock: เท่าที่ออกแบบเอาไว้คือจะหา testcase ที่จะทำการทดสอบว่า code ที่เราออกแบบไว้นั้นได้ใช้ในส่วนเราได้เขียนเอาไว้ตามกรณีที่เราคิดเอาไว้หรือไม่ ซึ่งในส่วนนี้จะมีความเกี่ยวข้องกับ Grammar มากพอสมควรทำให้คิดว่าการใช้เขียน test แบบ glassblock นั้นจะเอียงไปทาง Grammar ที่บอกไปในครั้งที่แล้วมากกว่า
Work plan

บอล 640610626	เป็นคนที่ทำงานในด้านของ blackend ทำให้ในส่วนของ Game State ก็จะยังคงเป็นคนออกแบบโครงสร้างเป็นหลัก และจะถามความคิดเห็นกับสมาชิกในกลุ่มที่เหลือว่า โอเคหรือไม่กับการออกแบบที่ได้คิดมาหรือไม่ อยากแก้อะไรตรงไหนรึเปล่า
ดิด 640610668	เป็นคนที่ทำงานในด้านของ frontend ทำให้ในส่วนของ Game State จะออกเป็นคนแสดงความคิดเห็นและลองอ่านดูว่ามีจุดไหนดูแปลกไปหรือไม่ โดยงานที่ทำในตอนนี้จึงเอียงไปทางด้านของ User Interface ที่เดินหน้าไปได้มากพอสมควร
พิมพ์ 640610654	เป็นคนออกแบบหน้าเว็บของตัวเกม UPBEAT ซึ่งจะเป็นคนที่ทำงานควบคู่ไปกับ ดิด ซึ่งก็จะถามความคิดเห็นกับทุกคนในกลุ่มว่าการออกแบบนี้ตรงตามที่ต้องการหรือไม่ หรือไม่ก็จะถามถึงส่วนที่จำเป็นต้องมีในหน้าเว็บ ทำให้ไม่ต่างจากคนที่ประสานการทำงานระหว่าง back กับ front

การเปลี่ยนแปลงหน้าที่ของคนในกลุ่มนั้นเกิดจาก การอธิบายการทำงาน code แล้วให้อีกฝ่ายไปเขียน test นั้น มีประสิทธิภาพการทำงานต่ำกว่าที่คิดเอาไว้ และในส่วนของทาง Frontend ก็มีปัญหาในเรื่องของไม่รู้ว่าจะออกแบบหน้าเว็บยังไงดี จึงเกิดการเปลี่ยนแปลงหน้าที่การทำงาน
และการออกแบบหน้าเว็บนั้น ทำให้ได้รู้ว่าควรมีข้อมูลจุดไหนที่ต้องการส่งไปยัง backend จึงทำให้เกิดการเชื่อต่อระหว่าง frontend และ backend มากยิ่งขึ้น
Known problems
1 css ทำงานไม่ตรงตามที่ต้องการ
2 map ไม่สามารถสร้างแบบที่คาดหวังได้ คือสร้างทีละ region เพราะตอนนี้สร้างได้ทีละ row แล้วเอามาต่อกัน
3 ยังนึกไม่ออกว่าจะ implement api จาก backend ยังไง
4 ยังไม่สามารถคิดได้ว่าการคำนวนเรื่องของคำสั่งต่างๆ นั้นต้องทำประาณไหน
5 จากการทำงานที่ผ่านมานั้นคิดว่า project นี้จะสามารถทำงานให้เสร็จได้ตามเวลาที่จารย์ชินได้กำหนดเอาไว้