We decided to use a Kanban methodology since it is agile thus compatible with our need to iteratively design and develop and has many advantages suited to our project. Three key reasons we chose this were the flexibility, iterative nature and ease of collaboration. The flexibility of dynamically assigning roles and work packages was crucial since all team members are full time students with other time commitments and clashing schedules. This meant most work had to be independently done outside of meetings, however this was not an issue as we used a Kanban board (Trello) to self-assign work packages as needed to meet deadlines for our deliverables. The Kanban board also allowed to add and assign priorities to each task and have clear visual indication of what stage each work package was in (to do, in progress, needs review, done) allowing for efficient remote collaboration between team members.

Due to the mostly remote working environment the team had decided on using an Instagram group chat to maintain a point of contact between all members of the team so any questions or concerns could be addressed. All members were already familiar with Instagram so using it was easy and accessible. Google meets were used to do conference calls when real time collaboration was needed or when role assignment and task assignment couldn't be done in person.

Development of the software was done using the LibGDX framework, which is an open-source Java toolkit for building cross-platform games from a single codebase. We used it because it provides unified APIs from graphics, input, audio and asset handling. Since none of us had prior game development experience using Java, its extensive supporting documentation and examples made it an appropriate choice, allowing us to learn it effectively and rapidly.

Version control of the software was managed with Git and was hosted on GitHub. Git provided a reliable history of changes, branching for parallel work when it was needed and safe merging of contributions. GitHub added a shared remote repository with pull requests and code reviews. This was effective for the group, and its web interface made it easy for the members who have not had experience with GitHub before.

For written documentation we used google docs to write all our supporting PDFs. It was easy for all team members to use and made it easy to implement any tables or diagrams into our documents. We considered using words however every member of the team has a university gmail account so setting up a shared private document for the whole group was much easier this way.

The group has a scheduled weekly meeting every Wednesday, these meetings were used to discuss the work allocation between members of the team and complete tasks assigned to them. Most of the plans had to be adjusted to meet deadlines as the amount of time the group would have to complete each package was more significantly hindered than anticipated by other uni work this meant the team had to use an iterative approach to our task. Additionally once development had begun group members would have to do risk monitoring at the end of their meetings this was effective as all members were present in these meetings so the discussions could cover the status of all risks identified.

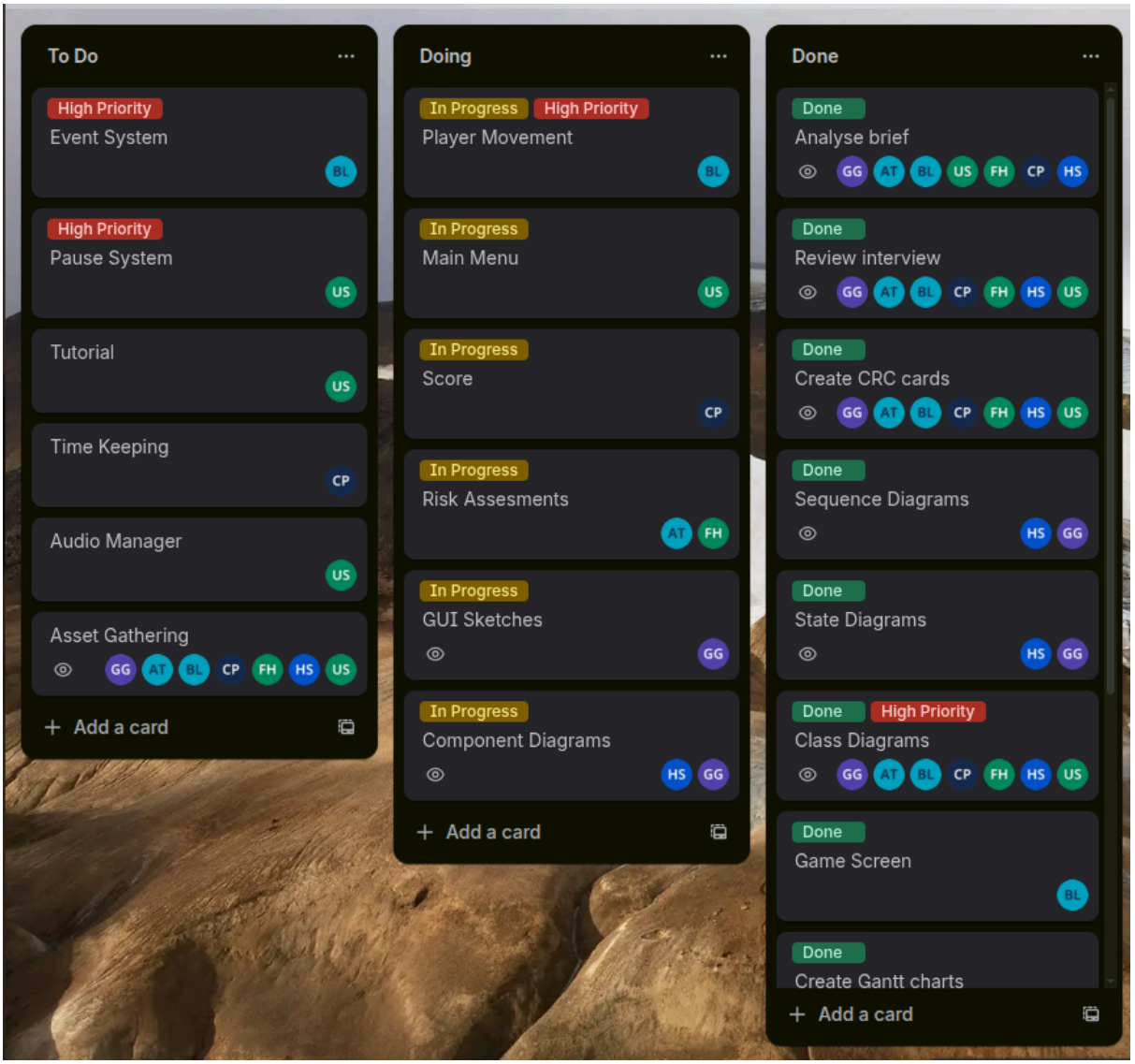
We assigned roles after completing our requirements elicitation and divided the work between development and design with members of the development team (Ben, Utkarsh, Chi) dividing the different work packages between themselves. While the development team started making the classes needed for the systems of the game the design team (Ali,Fares,Gal,Hux) created a comprehensive work breakdown diagram and created the first iteration of our risk register and set up the kanban board. The kanban board was an efficient way for the group to visually see the different priorities of each task and their statusTo do, doing or done.

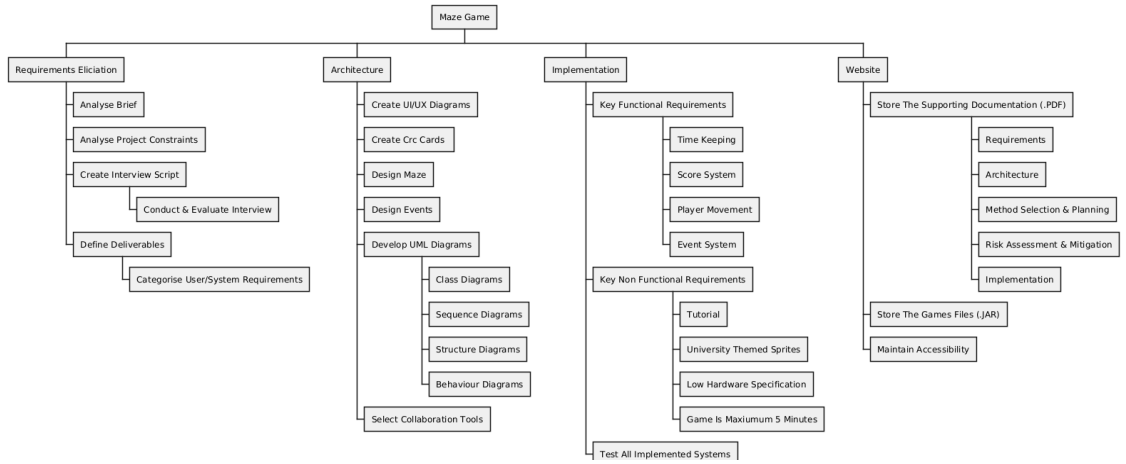
Work packages were put as a ticket on a shared kanban board. Each ticket was labelled with its priority and which team member has been assigned to the task. A larger Gantt chart was made which details the more holistic view of the work that needed to be done in the project that was used to estimate when tasks needed to be started by and when they should be finished along with detailing the dependencies of some of the systems needed to be developed for. This made it clear when tasks needed to be completed by and gave a clear linear development path for different sub systems of the game.

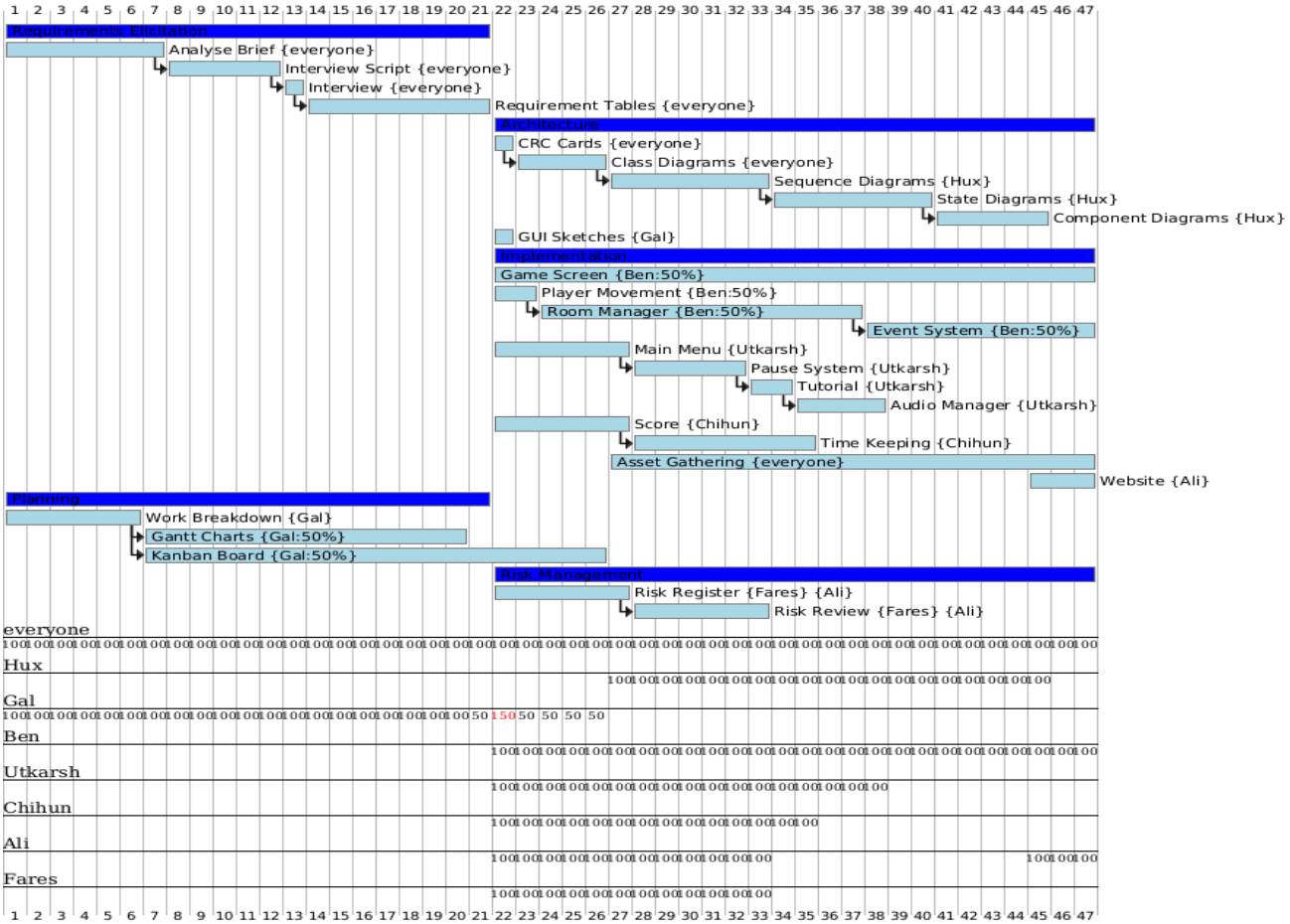
Below is the work breakdown diagram made using UML. It shows the necessary work packages needed to complete the game to the clients requirements. The architecture subtree decomposes the architecture of the game into simpler tasks like creating crc cards and developing UML diagrams, furthermore the develop UML diagrams task is split into sub tasks to create Class, Sequence, Structure and Behavior diagrams. The breakdown also splits the implementation of the game into the functional and non functional deliverables needed to meet the requirements.

Below is the gantt chart used to display the dependencies of each task and their necessary start and end date counted in days from the start of the project. The Gantt chart shows 5 main sections of the project that need to be completed in order to complete all the required deliverables requirements, planning, architecture, implementation and risk management.

Planning is broken down into creating the work breakdown structure and using that to develop a gantt chart to clearly visualize the dependencies between the designs that needed to be made and the implementation of the systems to create the game. Additionally we used a kanban board to clearly display the status of each work package and to assign task priorities using coloured labelling




Maze Game
Requirements Eliciation
Analyse Brief
Analyse Project Constraints
Create Interview Script
Conduct & Evaluate Interview
Define Deliverables
Categorise User/System Requirements
Architecture
Create UI/UX Diagrams
Create Crc Cards
Design Maze
Design Events
Develop UML Diagrams
Class Diagrams
Sequence Diagrams
Structure Diagrams
Behaviour Diagrams
Select Collaboration Tools
Implementation
Key Functional Requirements
Time Keeping
Score System
Player Movement
Event System
Key Non Functional Requirements
Tutorial
University Themed Sprites
Low Hardware Specification
Game Is Maxiumum 5 Minutes
Test All Implemented Systems
Website
Store The Supporting Documentation (.PDF)
Requirements
Architecture
Method Selection & Planning
Risk Assessment & Mitigation
Implementation
Store The Games Files (.JAR)
Maintain Accessibility


Requirements Elicitation
Analyse Brief {everyone}
Interview Script {everyone}
Interview {everyone}
Requirement Tables {everyone}
Architecture
CRC Cards {everyone}
Class Diagrams {everyone}
Sequence Diagrams {Hux}
State Diagrams {Hux}
Component Diagrams {Hux}
GUI Sketches {Gal}
Implementation
Game Screen {Ben:50%}
Player Movement {Ben:50%}
Room Manager {Ben:50%}
Event System {Ben:50%}
Main Menu {Utkarsh}
Pause System {Utkarsh}
Tutorial {Utkarsh}
Audio Manager {Utkarsh}
Score {Chihun}
Time Keeping {Chihun}
Asset Gathering {everyone}
Website {Ali}
Planning
Work Breakdown {Gal}
Gantt Charts {Gal:50%}
Kanban Board {Gal:50%}
Risk Management
Risk Register {Fares} {Ali}
Risk Review {Fares} {Ali}
everyone
Hux
Gal
Ben
Utkarsh
Chihun
Ali
Fares