UNIVERSITY OF THE WITWATERSRAND, JOHANNESBURG
SCHOOL OF COMPUTER SCIENCE & APPLIED MATHEMATICS

COMS3011A / COMS3028A
Software Design Project

Project Briefs & Rubrics

DRAFT: This is an incomplete document. The missing rubrics will be added over the first block of the semester

Contents

1 Overview

1.1 Key Requirements

Every group and their project must complete the following key requirements:

- Version Control - Students are expected to be using Git for code version control and an online repository tool such as Github to host their code.
- User Authentication & Security - Students must use independent authentication systems such as OAuth for security and secure usage of user information.
- Persistent Information - Students are expected to be building software that stores its state across different sessions and locations.
- Responsive and accessible UI - Students must build a user-interface that conforms to modern accessibility and responsiveness standards.
- Integration - Students are expected to be using an external API, and that their own API's are available for use by an external system.
- Testing - Students are expected to be testing their code in a variety of different ways: unit testing, integration testing, user testing etc.
- Continuous Improvement - Students are expected to follow a modern development methodology such as Agile or Scrum.
- Collaboration - Students should be collaborating within their own group, with any other groups that wish to use their API, and continuously meeting with their client to improve their product.

1.2 Selection Process

Each group is required to rank every project from one to ten in descending priority. For example, the project you would most like to work on is given rank 1, and the project you would like to work on the least is rank 10. Using a one-sided variant of the *Gale-Shapely* algorithm, each group will be assigned a project according to this ranking.

If your group would like to submit your own project idea, please fill in the questionnaire available on Moodle. This questionnaire should guide you on what information is required for your project to be allowed. It is better to add more information then less. A project may be rejected if it is considered to have insufficient work for your group to do over the semester, or if it does not align with the basic requirements all projects must have.

Each group is only allowed one project submission, and if it is rejected, you will be required to build one of the provided projects.

1.3 Timeline

The project will follow the below timeline of events:

Event	Description	End Date
Group Selection	Choosing project work groups	27 July
Project Assignments	Each groups particular project will be assigned	5 August
Sprint 1	Development starting: project setup and initial designs	19 August
Sprint 2	Development in progress: first features being implemented	2 September
Sprint 3	Development nearing completion: final features being implemented	29 September
Sprint 4	Project submission: feature complete and mostly bug-free	19 October
Documentation	A report about the decisions in the development cycle	19 October
Individual Reports	The individual reports are submitted	28 October
Presentations	The groups present their final product	22 - 28 October

2 Projects

2.1 Your own project

Please submit your answers to the following survey on the relevant Moodle questionnaire:

1. What is the name of your project?
2. Give a brief description of the project.
3. Provide a feature description of the project.
4. Provide a list of API modules your project will provide.
5. Provide a list of UI pages your project will requires.
6. Provide a list of database components your project requires.
7. Provide a potential use case of an external API.
   This does not need to be the final implemented external API.

2.2 Project 1: Campus Study Buddy

A platform that helps students find study partners, join study groups, schedule group sessions, and track topics they have covered. It is university community forum / social media to help students improve their experience with university work.
Features:

- Find study partners
  - Match with other students based on shared modules or topics.
  - View profile of students in your course
- Create study groups
  - Create and join group sessions.
  - Chat, share notes, and set group goals
- Track your progress
  - Log any completed topics, topics, and sections
  - Track and view your hours spent studying
- Plan sessions
  - Schedule future study session with reminders

UI Pages:

- Dashboard - Summary information, upcoming events, new groups etc.
- User Profiles
- Study Partner Search
- Group Session Planner

- Progress Tracker with tables and charts
- Chat window

API Modules:

- Partner Matching API - CRUD & Search on user preferences and profiles
- Group Session API - CRUD for groups, members, and scheduling
- Progress API - CRUD for topics, completion state, and hour tracking
- Notifications API - Manage reminders and updates

Database Entities:

- User profiles
- Study groups
- Topics, chapters, other material
- Notifications & reminders

2.3 Project 2: Campus Quest

A location-based game that encourages students to explore the university campus by completing quests, finding hidden markers, and earning rewards.
Features:

- Location Quests
  - Complete quests by visiting specific campus buildings or landmarks.
- Treasure Hunts
  - Unlock clues to hidden markers placed around campus.
- Collectibles
  - Earn digital badges, items, and collectibles for achievements.
- Leaderboard
  - Compete with classmates for top scores each week.

API Modules:

- Quest API – CRUD for quests and challenges.
- Collectibles API – CRUD for badges and rewards.
- Leaderboard API – Track scores and rankings.
- Location API – Verify user location for quest progress.

UI Modules:

- Campus Map with quest markers

- Active Quests list
- Leaderboard view
- Profile & Inventory page

Database Components:

- Quests and their locations
- Collectibles inventory
- User profiles and progress
- Leaderboard scores

2.4 Project 3: Event Planning

Build a website that helps users prepare and manage all aspects of an upcoming event, including guest invitations, vendor bookings, and event design.
Features:

- Guest Management
  - Create guest list, send invites, track RSVPs manually.
- Vendor & Venue Planning
  - Add and manage venues, vendors, and booking notes locally.
  - Compare manually entered prices and availability.
- Event Blueprint
  - Manually input schedule, session details, and floorplans.
  - Export as downloadable files or printable materials.
  - Provide schedule, guests, and layout in standardized JSON or CSV format.

API Modules:

- GuestList API – Add, update, and retrieve guest details and RSVP status.
- Vendor API – Manage venues, catering, and other services.
- EventData API – Event metadata (theme, date, schedule, maps).
- Export API – Provide downloadable event packages for other apps.

UI Modules:

- Guest Manager
  - View RSVPs, invite guests, group by tags (e.g., VIP, Dietary).
- Vendor Dashboard
  - Add vendors, link contracts, track costs.

- Event Overview
  - Editable schedule and theme setup.
  - Upload/download documents (e.g., seating chart).
- Integration Tools
  - Export finalized event data for external use.

Database Components:

- Guests – Name, contact, RSVP status, dietary info.
- Vendors – Type, name, cost, status, notes.
- Event – Name, date/time, theme, host info.
- Files – Uploaded layouts, contracts, and exports.

2.5 Project 4: Event Management

Build a website that supports real-time event management on the day of the event — guest check-ins, schedule monitoring, and live updates.
Features:

- Guest Check-In
  - Check guests in via manual search or built-in QR generation.
  - Upload guest lists manually if integration is unavailable.
- Schedule & Announcements
  - View and edit a real-time schedule for events or sessions.
  - Push announcements to displays or message boards.
- Interactive Maps
  - Display a manually uploaded floorplan and mark points of interest.
  - Use drawing tools to annotate in real-time.
- On-Site Coordination
  - Assign tasks to staff, log incidents, and mark task completion.
  - Offline support for mobile teams.

API Modules:

- Check-In API – Update and view guest entry data.
- Schedule API – Modify and retrieve live session data.
- Map API – Serve event layout and allow annotations.
- Staff API – Handle task assignments, completion, and internal logs.

UI Modules:

- Check-In Console
  - Scan or search guests, manually mark arrivals.
- Live Schedule View
  - Editable agenda, broadcast announcements.
- Venue Map
  - Upload static images of floorplans.
  - Add labels or emergency notes.
- Staff Dashboard
  - Assign staff roles, log incidents, real-time team chat.

Database Components:

- Guest Records – Name, check-in status, optional QR code.
- Schedule Items – Time, location, description, status.
- Venue Maps – Uploaded images, interactive overlays.
- Staff Logs – Assigned tasks, completion status, messages.

2.6 Project 5: Sport Stat Tracker

Create a stat-tracking tool for team or individual sports. It allows users (coaches, analysts, or fans) to record game stats in real-time or after matches. It tracks players, performance, and historical records, and a host of other information.
Features:

- Player & Team Management
  - Add teams, players, and assign roles or positions.
  - View historical stats for each player or team.
- Post-Event Tracking
  - Input goals, fouls, assists, shots, possession, or custom stat types.
  - Supports uploading of standard format.
- Stat Reports
  - Generate summaries of game, player, or team performance.
  - Compare stats over a season or tournament.

API Modules:

- Team API – Add/edit players, teams, and rosters.
- Match Stats API – CRUD for game-specific stats.
- Player History API – View aggregated performance by player/team.

UI Modules:

- Team Manager – Create teams and assign players.
- Match Entry – Interface for adding stat events during or after a game.
- Stat Dashboard – View game summaries, top players, and team comparisons.
- Export Tools – Export stats to CSV, PDF or JSON.

Database Components:

- Teams – Team name, roster, coach, colours.
- Players – Name, team ID, position, jersey number.
- Match Data – Game ID, stats by timestamp, opponent, score.
- Player Stats – Aggregated stats by player, game, and season.

2.7 Project 6: Sport Live Feeds

A real-time sports broadcasting and viewer experience tool. It provides live updates, visualizations, and commentary feeds for ongoing games.
Features:

- Match Viewer
  - View current score, game clock, possession, and key events (goals, fouls, etc.).
  - Display structured data for fans, commentators, or organizers.
- Event Feed
  - View live timeline of in-game events with timestamps and descriptions.
  - Animate key actions like substitutions or cards.
- Match Setup
  - Manually create matches and add teams, players, and expected schedule.
- Live Input
  - Manually input match events and score changes if not integrated with a live tools.
  - Supports pause/resume and timeline edits.

API Modules:

- Live Update API – Receive and store stat events from external sources.
- Feed API – Retrieve current state of the match for front-end display.
- Match Setup API – CRUD for game meta (teams, match time, venue).
- Display API – Serve structured game data to clients (scoreboards, overlays, etc.)

UI Modules:

- Live Scoreboard – Displays current score, teams, and game status.

- Event Timeline – Chronological list of major match events.
- Match Setup Screen – Add new matches, assign teams and times.
- Manual Input Panel – Add stat events like goals, substitutions, cards manually.

Database Entities:

- Match Info – Match ID, start time, team IDs, current status.
- Event Log – Timestamped stat events (e.g., goal, foul, timeout).
- Display State – Current score, clock, possession, game phase.
- Player & Team Data – Synced or manually entered team rosters.

2.8 Project 7: Hiking Logbook

Tracking completed hikes was historically done with a physical logbook. Hiking Log serves the same role, but available on the web. Track your hikes, their location, distance and thoughts, on the web anywhere.
Features:

- Track Your Hikes
  - Keep notes on the location, weather, elevation, and route.
  - Simple notes & thoughts: editable on page.
  - Keep track of your accomplishments along the way.
- Plan Your Hikes
  - Select routes, view weather on the day, choose start times.
  - Invite friends to join planned hikes.
  - Autolog a hike upon completion.
- View Friends' Hikes
  - Feed of friends' activity.
  - See recent hikes and shared goals.
  - Share victories and milestones with friends.

API Modules

- Logbook API – Logbook record CRUD; query by hike info.
- Friends API – Fetch, view, and interact with user friends.
- Achievements API – Handle accomplishments and goals; derive from hike stats (distance, time, days).
- Planning API – Manage planned hikes, equipment checklists, invited friends.

UI Pages

- Logbook – View, update, edit logbook; filter/search; view hike details including route map, elevation, and time data.
- Activity Feed – View recent activity and achievements from friends and self.

- Hike Planner – Create, view, and edit planned hikes; invite friends; leave hike notes.
- Achievements – Create/view/edit goals; pin hikes; visualize progress with charts.

Database Entities

- Logbook Data – Record of hikes: route, name, location, difficulty, weather, start/end date and time, duration, GPS waypoints.
- Friend Data – Connected accounts; joint hike records.
- Achievements Data – Custom user goals, progress, pinned hikes.
- Planned Hikes – Route, name, date, invite list, and related metadata.

2.9 Project 8: Crowdsource Hiking Information

A centralized, crowd-sourced hiking trail database. It provides up-to-date information on hiking trails across the country, including difficulty, terrain, photos, and user reviews. Designed to help hikers discover, plan, and prepare for outdoor adventures.
Features:

- Browse Trails
  - Search by location, difficulty, distance, and terrain.
  - View trail descriptions, photos, reviews, and maps.
- Community Contributions
  - Users can submit new trails or update information on existing ones.
  - Add photos, rate difficulty, and write reviews.
- Favourites and Wishlists
  - Save trails for later.
  - Mark completed trails and maintain a wishlist of hikes.
- Safety & Alerts
  - View trail closures, conditions, or alerts from the community or authorities.
  - Sign up for notifications on saved trails.

API Modules

- Trail API – CRUD for trail data including name, location, description, and difficulty.
- User Contribution API – Submit photos, reviews, updates.
- Search & Filter API – Location-based and parameter-based search over trail listings.
- Favourites API – Manage saved and completed trail lists for a user.
- Alerts API – Add and fetch status updates on trail conditions.

UI Pages

- Trail Explorer – Search, view, and filter trails on a map or list; view trail detail pages.

- Trail Submission – Submit a new trail or edit existing trail info.
- Reviews and Media – Submit and view photos, reviews, ratings.
- My Trails – Manage favorites, completed hikes, and wishlist.
- Alerts & Updates – Subscribe to trail alerts; view community updates and warnings.

Database Entities

- Trail Data – Name, location, distance, elevation gain, difficulty, tags, GPS route.
- User Data – Profile info, submitted trails, reviews, favorites.
- Reviews and Media – Trail reviews, ratings, uploaded photos.
- Alerts – Community- or system-generated notifications tied to specific trails.

2.10 Project 9: Virtual Pen Pals

GlobeTalk connects users anonymously with random people around the world for friendly, cultural message exchanges. The app simulates the experience of having a pen pal — but through modern digital messaging with a strong focus on curiosity, connection, and global diversity.

Feature Description

- Random Matchmaking:
  - Match with a new pen pal from a random country based on time zone or language preferences.
  - Choose between one-time messages or long-term correspondence.
- Asynchronous Messaging:
  - Send and receive text-based letters with a delivery delay (e.g., 12 hours) to simulate postal mail.
  - Messages are stored privately and can include basic emojis.
- Cultural Profiles:
  - Users can write a short intro about themselves and read anonymous facts about others (age range, hobbies, region).
  - Includes region-based fun facts, holidays, or sayings.
- Safety & Moderation:
  - Flag inappropriate content, block users, and keep all chats anonymous (no names or emails).
  - No media or file-sharing — text only.

API Modules

- Matchmaking API – Connect users with available pen pals using filters.
- Message API – Store, delay, and retrieve messages between matched users.
- Profile API – Handle cultural profile info (region, hobbies, languages).
- Moderation API – Report users, manage bans, and review flagged content.

UI Modules

- Match Screen – Find a new pen pal with optional filters.
- Message Inbox – View active conversations and reply to letters.
- Compose Letter – Write and send delayed text-based messages.
- Cultural Explorer – View country-based facts or info about your pen pal's region.
- Settings & Safety – Block/report users, adjust preferences, or view app policies.

Database Components

- User Profiles – Language preferences, interests, region (approximate), and basic info.
- Match Records – Pairings of users with timestamps and message thread links.
- Message Storage – Message content, delivery timestamps, sender/receiver IDs.
- Moderation Logs – Flags, reports, bans, and moderator decisions.

2.11 Project 10: Open-source Language Learning

A community-driven language learning platform where anyone can create, upload, and share language courses. It supports vocabulary drills, grammar lessons, and pronunciation guides, allowing learners to study any language — even rare or constructed ones — in a collaborative, open-source environment.
Features:

- Course Creation
  - Users can author language courses using custom lesson formats (text, multiple choice, audio, etc.).
  - Lessons are grouped into units with progress tracking.
- Learner Dashboard
  - Track progress through lessons, view scores, and revisit difficult topics.
  - Set learning goals and receive daily reminders.
- Review & Quiz Engine
  - Built-in flashcard and quiz tools with spaced repetition.
  - Auto-generated reviews based on user performance.
- Explore Courses
  - Browse or search courses by language, popularity, difficulty, or tags.
  - Preview course content before enrolling.
- Community Contributions
  - Rate courses, leave feedback, and suggest edits to open content.
  - Upvote helpful explanations or translations.

API Modules

- Course API – CRUD operations for courses, units, and lessons.

- User Progress API – Track scores, lesson completion, and streaks.
- Quiz Engine API – Deliver and score quizzes and flashcards.
- Community API – Ratings, comments, edits, and moderation for courses.

UI Pages

- Course Explorer – Browse/search for courses and preview content.
- Lesson Player – Interactive lesson interface with text, images, and quizzes.
- Course Builder – Upload or edit courses with markdown-style formatting and lesson templates.
- Review Dashboard – Personalized recap of difficult vocabulary, grammar, and past scores.
- User Profile – Track learning progress, created courses, and contributions.

Database Entities

- Courses – Title, description, language, tags, author, difficulty level.
- Lessons – Linked to courses, includes content type, structure, and media.
- User Progress – Tracks completed lessons, quiz scores, streaks.
- Community Feedback – Ratings, comments, upvotes, suggested edits.
- Media Storage – Audio recordings, images, or supplementary files.

3 Rubrics

All rubric criteria use the following breakdown of marks:

1. Exceeded Expectations (EE): 80% – 100%
2. Met Expectations (ME): 50% – 80%
3. Below Expectations (BE): 0% – 50%

3.1 Sprint 1

Criteria	**Weight**	**EE (80% – 100%)**	**ME (50% – 80%)**	**BE (0% – 50%)**
Version Control	10%	Exists – code quality maintenance tools used (linters, package managers, bundlers etc.) Used – all members have committed code	Exists – organised file structure & readme Used – most members are committing code	No Git repository Local Git repository – organised – readme Online repository – unorganised – readme – more than one member has committed code
Documentation Site	10%	Well designed Has non-trivial content	Has trivial content – repository links / member info / project briefs etc.	No documentation aggregation Local documentation aggregation Online but inaccessible aggregation Version-controlled documentation Online and accessible
Project Work Tracker	5%	Exists & Used	Exists	Does not exist
Development Guides	5%	How to create a development database	How to create a development API	How to create a development site
Git Methodology	5%	Documentation & Resources explaining methodology	Pre-existing methodology / clearly documented methodology	No methodology / Insufficient methodology
Project Management Methodology	15%	Evidence of followed procedure Documentation on rationale	Methodology references & documentation Detailed methodology procedure	No methodology / Invented methodology / Listed but unexplained methodology
Technology Stack	5%	Tech stack fully listed & explained	Majority of tech stack components are decided	No to few tech stack components are decided on
Stakeholder Interaction	10%	Have identified all hidden requirements	Have identified all unhidden requirements	None to poor identification of features
Initial Design & Development Plan	15%	Comprehensive development plan & roadmap	Adequate development plan	Incomplete development plan
Implementation	15%	Authentication & other supporting features implemented	Supporting features being implemented	No or severely broken / faked implementations

3.2 Sprint 2

Criteria	**Weight**	**EE**	**ME**	**BE**
Core Features	25%	Implemented, at most one non-severe bugs	implemented with non-severe bugs	not implemented / unfinished implementation / implemented with severe bugs
Automated Testing	10%	UI & API Testing implemented – no flakey tests, useful tests	UI or API testing implemented – simple testing routines	No testing
Stakeholder reviews	10%	Regular stakeholder meetings & stakeholder feedback evaluated & implemented	Occasional stakeholder meetings & stakeholder feedback evaluated	no stakeholder meetings / stakeholder feedback ignored
API Integration	15%	Implemented external API usage & project API fully available for external usage	mostly implemented external API usage / Project API mostly available for external usage	No / minimal external API integration / Project API not available for external usage
User Feedback	10%	Extensive user testing & formal feedback collection process & evidence of feedback integration	Some user testing – formal feedback collection process & evidence of planned feedback integration	None to little user testing / informal collection process
Project Management Methodology	10%	Actively following methodology	mostly following methodology	none or poor following of methodology
Bug Tracker	5%	Extensive use of a bug tracker	Use of a bug tracker	no use of a bug tracker
Database Documentation	5%	Full database schema & deployment info & choice justification	database schema & deployment info	
Third-Party Code Documentation	5%	All third-party code documented & justified (important third-party code)	Most third-party code documented & some justified	None to minimal documentation
Testing Documentation	5%	Extensive documentation on testing – user feedback formal process & how to use automated tests	Documentation on most testing – user feedback process / how to use automated tests	none to minimal documentation

3.3 Sprint 3

Criteria	**Weight**	**EE**	**ME**	**BE**
Feedback	10%	Extensive user testing and feedback gathering	Some user testing and feedback gathering	No user testing
Automated Testing	10%	Extensive automated testing across all code bases (code coverage $\geq$ 80%)	Extensive testing in one code base / Middling testing of all code bases ($\geq$ 50%)	No automated testing / Low testing across all codebases ($\leq$ 20%) / middling testing in one code base
Feature Implementation	20%	Most features implemented with minimal bugs / All features implemented with non-severe bugs	Most features with non-severe bugs / All features implemented with bugs	No additional features from sprint 2 / Some additional features with bugs / Most features implemented with bugs
API Implementation	20%	All endpoints implemented mostly bug-free	All endpoints implemented with non-severe bugs	Not available for external usage / Some endpoints implemented / Most endpoints implemented with bugs
Performance	5%	No performance issues	Minimal performance issues	Large performance issues
Improvement	5%	Extensive evidence of integrating user and stakeholder feedback	Some evidence of integrating user and stakeholder feedback	No evidence of integrating user & stakeholder feedback
Documentation	15%	All aspects (features / API / database / testing etc.) of the project are well documented	Most aspects of the project are well documented	No aspects of the project are documented / Some aspects of the project are documented / Most aspects of the project are documented
Methodology	15%	Evidence of strictly following methodology	Evidence of mostly following methodology	no or poor evidence of following methodology

3.4 Sprint 4 - Final Submission

3.4.1 Database

Criteria	**Weight**	**Description**
Data	3%	Do you have production data? How much of your data is from testing?
Deployment	2%	Database is deployed
Structure	5%	How well structured is your Database? Are you following best practice for your database type?

3.4.2 API

Criteria	**Weight**	**Description**
Availability	3%	Is your API available for external and internal usage.
Architecture	5%	How well your API follows an established API architecture and all of its principles
Deployment	2%	API is deployed
Performance	5%	How well does your API handle significant loads, how often does it crash, and how well does it handle incorrectly formatted requests etc.
Design	10%	Does your API make use of the different HTTP methods correctly? Do you have redundant endpoints? Does the organization & hierarchy make sense to an external user?

3.4.3 Web-App

Criteria	**Weight**	**Description**
Accessibility	5%	Is your product accessible? Contrast-Ratio's, use of semantic HTML or ARIA etc.
Aesthetics	3%	Does your product look good? Does it have consistent styling?
User Experience	5%	How do you handle errors? How long does it take a user to use some feature? Can the user re-open a closed tab and continue where they left off or do they need to re-login?
Deployment	2%	Web-App is deployed
Performance	5%	How long does it take to load? How responsive is the program to input?
Features	10%	How well implemented are your features? Does your product do what your client intended it to do?
Responsiveness	5%	How responsive is your UI? Is it mobile friendly? Does it change according to screen size?
Structure	5%	How complex is the interface? How difficult is it to navigate to a new feature? Does it make sense to a first time user?

3.4.4 Misc

Criteria	**Weight**	**Description**
Git Methodology	3%	How well did you follow your chosen git methodology?
Integration	7%	Are you integrating with no external API's, a separate API or another group's API?
Testing	10%	How well written are your tests? Code Coverage ($\geq 80\%$, $\geq 50\%$)? What are you *not* testing?
Tools	5%	Did you use a project work tracker? & Did you use a bug tracker? & What code quality tools did you use, and how did you enforce them?

3.5 Group Report / Documentation

Criteria	**Mark**	**Description**
Introduction	10	
Methodology & Planning	30	Choice of Development Methodology & Following of Development Methodology & Review of Development Methodology & Project Design & Project Planning & Stakeholder Interaction
Architecture	20	API Documentation & Database Documentation & Developer Guides & Tech Stack & Third-Party Code Documentation & User Guides
Testing	20	Automated Testing Procedure & User Feedback Procedure & User Feedback Analysis and Review
Deployment & Integration	20	Deployment Platform Choice & Deployment Platform Review & Integration Review
General Discussion	20	Challenges & Future Development & Project Evaluation
Conclusion	5	
Supporting Documents	5	An appendix of the documents you use to substantiate your arguments in the report.
Total	130	

3.6 Individual Report

Criteria	**Mark**	**Description**
Introduction	10	
Responsibilities & Contributions	20	What design elements were your responsible for? & How successful were your contributions? & Justify your material impact on the project.
Technical Challenges & Solutions	20	What technical problems did *you* encounter in the development lifecycle? & How might you pre-empt these problems in the future? & What was the worst technical decision made in your project? & What was the best technical decision made in the your project?
Learning & Development	20	What AI tools did you use, and what is the underlying model for the tool (GPT, LLama, Claude)? & How did you primarily interact with AI? & What is your opinion on the use of AI in software development? & What other new (to you) technology did you use in the project? & What technology do you *think* might have been useful, but did not use?
Collaboration & Teamwork	10	How well did your group work together? & What social collaboration techniques did you use in your group (blind-voting, brainstorming sessions etc.)? & Discuss the integration process with the other groups, or if you did not integrate with a group, why?
General Discussion	10	What is something that you did that you would like to discuss? & What is your general opinion (and why) on the project
Conclusion	5	
Supporting Documents	5	An appendix of the documents you use to substantiate your arguments in the report.
Total	100	

3.7 Presentation

Criteria	**Mark**	**Description**
Product Showcase	40	Product Overview & Feature Demonstrations & User Interface and Experience
Technical Breakdown	20	Architecture Overview & Functionality Implementation High-level Descriptions
Presentation Quality	40	Structure & Time Usage & Product / Project Coverage & Visual Aids & Team Collaboration
Questions	10	How well do you respond to questions (understanding of the question and comprehensive responses)? & Team Collaboration on question responses
Total	110	