IBM TODO App Project

Architectural Decisions

Software Engineering Project (CSU33013 & CSU22013)

10th March 2022

Brian Whelan (3rd year)

Adriana Hrabowych (3rd year)

Tom Roberts (3rd year)

Nadia Abouelleil (3rd year)

Arshad Rehman Mohammed (2nd year)

James Merrins Pryce (2nd year)

Liam Reilly (2nd year)

Architectural Decisions

Some of the architectural decisions were made for us by the client in the project specification but a number of decisions had to be made by the team as we encountered certain issues. Through consultation with the client, it was advised to document these decisions to ensure that we were considering all possible options and justifying our final architectural design.

Below are the architectural decisions we encountered over the course of the project, the solutions we considered and the ultimate decision we made to solve the issue. These architectural decisions have been created using the architecture decision description template published in "Architecture Decisions: Demystifying Architecture" by Jeff Tyree and Art Akerman, Capital One Financial. These templates are active documents and will be updated as and when architectural decisions are made or changed over the course of the project - the most up-to-date version of this document can be found in the repository, where they will be updated as decisions are made/changed.

Issue	The project requires collaboration and constant development of an application, and therefore it must be possible to easily make and revert changes in the codebase.
Decision	Use a version control system, namely Git, to track changes and versions of the codebase to enable backtracking.
Status	Decided
Group	Project Management
Assumptions	● We cannot guarantee asynchronous development amongst the team
Constraints	None
Positions	● Using SVN version control
Argument	Git version control is distributed, meaning that we can each have a local copy of the repository, make changes and commits and then push to the remote repository when we have a connection. This enables work to be done even when offline. In contrast, SVN is centralized, meaning commits cannot be made locally and require a connection.
Implications	The team will need to download Git onto their local machine to enable the creation and cloning of repositories.
Related Decisions	None

Related Requirements	None
Related Artifacts	None
Related Principles	None
Notes	• All of the team has some minor experience with SVN from 1st year programming module • Those who have experienced Git have found it much more user friendly and easy-to-use • Git is the most popular version control system and would be useful to have knowledge of going forward

Issue	The project requires a well-defined backlog to be maintained to ensure each team member is clear on what has been done, is being done and needs to be done at every stage.
Decision	Use a Kanban board, namely GitHub Project Boards, to define the project backlog, assign tasks, and monitor the progress of the project.
Status	Decided
Group	Project Management
Assumptions	• Managing the project backlog through scrum meetings alone is not viable
Constraints	None
Positions	• Using Trello
Argument	As we will be using GitHub, using GitHub Project Boards offers a seamless integration of the kanban board with the remote repository. Using a separate kanban board tool such as Trello would have required someone to maintain the board regularly as opposed to having the board update dynamically as tasks are completed within the repository.
Implications	The team will need to learn how to make use of the Project Boards feature by creating cards, assigning them and updating their progress.
Related Decisions	None
Related Requirements	None
Related Artifacts	None
Related Principles	None

Notes	● Some of the team has used Trello before, however GitHub Project Boards offers a more transparent kanban board feature and is integrated in the GitHub ecosystem

Issue	The project requires the automation of software development workflows, namely automating the integration and deployment of the application.
Decision	Use a platform, namely GitHub Actions, that enables the automation of the build, test and deployment pipeline
Status	Decided
Group	CI/CD Pipeline
Assumptions	None
Constraints	None
Positions	● Using Jenkins
Argument	As we will be using GitHub, GitHub Actions offers an integrated platform with the GitHub ecosystem. The learning curve for GitHub Actions also seems smaller than that of Jenkins which is key in the short time of the project.
Implications	The team will need to become familiar with creating workflows using GitHub Actions and customising them to the project requirements.
Related Decisions	None
Related Requirements	None
Related Artifacts	None
Related Principles	None
Notes	● Very little knowledge of automation tools such as GitHub Actions or Jenkins amongst the team ● Learning curve of GitHub actions seems relatively smaller than that of Jenkins

Issue	The project requires that the application data be stored in a persistent database
Decision	Use MongoDB to store and retrieve application data
Status	Pending

Group	Backend
Assumptions	• The data must be easily accessible
Constraints	None
Positions	• Using MySQL • Using SQLite • Using PostgreSQL
Argument	
Implications	The backend team will need to become familiar with the MongoDB API for storage and retrieval of data, as well as configuring the database as required.
Related Decisions	None
Related Requirements	None
Related Artifacts	None
Related Principles	None
Notes	

Issue	The project will be completed over a number of months and as such it is essential that all additions can be verified to work as intended to prevent any unnecessary blockages.
Decision	Use a JavaScript unit testing framework, namely Jest, to write unit tests for new functions and features.
Status	Decided
Group	Backend
Assumptions	• Building a substantial application is not feasible if bugs cannot be easily found and rectified
Constraints	None
Positions	• Using Mocha • Using Cypress • Using Webdriver
Argument	Jest was designed by Facebook developers to work with React and so it fits seamlessly with our React application. Jest is also relatively simple and requires no pre-configuration unlike Mocha. Jest is also specifically designed for unit testing as opposed to more comprehensive testing frameworks like Cypress and Webdriver which offer a variety of additional

	testing mechanisms which are not required for the purposes of this project.
Implications	The backend team will need to become familiar with the syntax and write Jest unit tests when developing new features and functions.
Related Decisions	None
Related Requirements	None
Related Artifacts	None
Related Principles	None
Notes	● The team all has some knowledge of unit testing and have all used JUnit to test Java applications ● Using JavaScript itself is new to the majority of the team aso using a simple unit testing framework would be preferable ● As we are using React, it seemed obvious to use Jest given how closely linked the two are

Issue	The project will require a frontend UI for the application that is both functional and visually appealing.
Decision	Use JavaScript and moreover use the React library to develop the application frontend
Status	Decided
Group	Frontend
Assumptions	None
Constraints	None
Positions	● Using vanilla JavaScript ● Using Vue.js
Argument	Using React removes many of the complexities of working vanilla JavaScript and greatly simplifies the development of the application. React is also more catered toward cross-platform as opposed to Vue.js and while this application will not be cross-platform for this project, given the nature of the application, it may be something to consider for future development.
Implications	The frontend team will need to familiarise itself with React and JavaScript in general.

Related Decisions	• Choice of unit testing framework
Related Requirements	None
Related Artifacts	None
Related Principles	None
Notes	• JavaScript offers the most comprehensive and common tools and frameworks for developing frontend applications • Some of the team members have some basic understanding of JavaScript and some have some experience with React

Issue	The project require a backend framework to build the application
Decision	Use Node.js
Status	Decided
Group	Backend
Assumptions	None
Constraints	None
Positions	• Use Java
Argument	Using Node.js integrates well with our frontend React application and also makes integration with a database relatively straightforward.
Implications	The backend team will have to learn Node.js.
Related Decisions	• Choice of frontend • Choice of unit testing framework • Choice of database
Related Requirements	None
Related Artifacts	None
Related Principles	None
Notes	• The team has extensive knowledge of Java from college already • Using JavaScript and Node.js might be best for both learning and also for the purposes of the application

	• The JavaScript syntax is relatively similar to that of Java and so should be relatively familiar

Issue	The project requires some static analysis of the codebase to assess code quality, detect bugs and detect vulnerabilities
Decision	Use Sonarqube to perform static analysis of the codebase
Status	Pending
Group	CI/CD Pipeline
Assumptions	• The quality and security of code cannot be left solely to developers
Constraints	None
Positions	• Using Sonarqube • Using pre-commit • Using snyk
Argument	Sonarqube is an industry standard and probably the most valuable static analysis tool to use
Implications	The team will have to become familiar with Sonarqube and static analysis
Related Decisions	None
Related Requirements	None
Related Artifacts	None
Related Principles	None
Notes	• Sonarqube is an industry standard and probably the most valuable static analysis tool to use