Fullstack Development Bootcamp - JavaScript

Capstone Project

Guidebook

**Introduction**

Congratulations on reaching the end of the full-stack development bootcamp. Along the way you learnt a ton of concepts, workflows and techniques that enable you to build frontend applications, APIs, work with databases and finally culminate these learnings into building full-stack applications.

To complete this experience, it is important to put your learnings into a well-defined full-stack project. This project is part of the mandatory requirements for successfully completing the program and is a gradable activity.

This document offers important points regarding your capstone project. While you're free to bring your own idea and submit that as a project, we have on our part, provided you with several outlines that you may pick from to build your project.

Please reach out to your buddy or learning advisor if you have any questions.

Happy coding!

The upGrad KnowledgeHut Team

**Key points to consider before submitting your project**

When we evaluate a full-stack capstone project, we're looking for the following attributes. Therefore, it is important for you to keep these in mind while building your full-stack project for submission:

1. **Functionality:** Does the project deliver the required functionality and meet the requirements? Are all the features correctly built and functioning as they should?
2. **User experience**: Is the project easy to use and navigate? Is the user interface intuitive and visually appealing?
3. **Code quality**: Is the code well-written and organized? Is it easy to read and understand? Is it free of syntax errors and other bugs?
4. **Testing**: Has the project been thoroughly tested to ensure that all the features are working correctly?
5. **Documentation**: Is the project well-documented, with clear instructions for how to install, set up, and use the application?
6. **Deployment**: Is the project successfully deployed and accessible to users?
7. **Attention to Detail**: This is probably the most important of all. When you create something, the attention to detail that you put into the project spells your ability to handle work seriously. This is an indirect indicator of your attitude towards work. Whether it is correctly formatting code, naming files, documentation to visual details and feature design, attention to detail is critical.

Overall, a good full stack capstone project should be functional, user-friendly, well-written, well-tested, well-documented, and deployed successfully.

While hunting for an idea to build as a MERN stack project, you should keep the following points in mind:

1. Define a well-defined and interesting problem to solve. The problem should be relevant to the MERN stack, and should be solvable using the technologies in the stack.
2. Set clear and achievable goals and objectives. The goals and objectives should be specific, measurable, attainable, relevant, and time-bound (SMART).
3. Design a user-friendly and intuitive interface. The interface should be easy to navigate and use, and should provide a pleasant and seamless user experience.
4. Write clear and comprehensive documentation. The documentation should include instructions for setting up and running the project, as well as detailed information about the project's features and functionality.
5. Ensure scalability and extensibility. The project should be designed in a way that makes it easy to expand or modify in response to changing user needs or requirements.
6. Test the project thoroughly. The project should be tested extensively to ensure that it functions correctly and reliably. This should include both functional and non-functional testing, as well as testing for security and performance.
7. Deploy the project successfully. The project should be deployed to a hosting platform, such as Heroku or AWS, and should be accessible to users.
8. Maintain and update the project regularly. The project should be maintained and updated over time to fix bugs, add new features, and improve performance.

**Capstone Project - Features**

Your capstone project isn't just meant to signal the end of your Fullstack program, it is above everything else, meant to be an example and showcase of your abilities as a Fullstack developer. This is why, it must be dealt with care, focus and with an eye for detail. And while we officially require one project to be submitted as a part of the program completion criteria, we encourage you to build as many Fullstack projects as you can, as part of your personal portfolio. These go a long way in assuring potential employers on the measure of your abilities and it naturally translates to better employment opportunities.

This brings us to the question - What kind of features should you include in your capstone project?

Here are some of the features that you should ideally include in your Fullstack capstone project. While it is not mandatory to include everything, this list serves as a good measure of the extent to which you can go to ensure favourable outcomes especially when approaching employment opportunities.

* **The Frontend:** Build an appealing frontend application using React, React Router and optionally include Redux (only if needed. You may alternatively ). This application will be the first thing that an evaluator or observer will witness, hence give it your best shot. Build an impressive user interface. You can look up sites like dribbble.com for inspiration. Alternatively, if you're building a clone of an existing product/service, spend time on observing their UI, interactions and features and try to create something similar. Add your own flair to make it interesting and feel free to tone down and simplify features.
* **The Backend:** Your frontend application will require APIs. While there are several ways to create a backend today (such as using Google's Firebase, Supabase etc.), we encourage you to build-your-own-backend from scratch using the tools you've learnt - Node.js, Express. While this may be time consuming at first, it will give you a first-hand experience of the process and will only build your proficiency and expertise which you ultimately require in your career. Do not be shy from taking this challenge head-on. Your Express based backend must expose API endpoints and they should be documented properly.   
    
  Another important consideration when building the backend using Express is the ability to serve the static frontend application that you will build using React. While in production this isn't an advised practice because the backend in this case, an API server must only expose APIs and nothing else, in the case of your project, you want the application to be a monolith and something that can be deployed together. So, ideally, expose your APIs on the /api base route while the root route (/) should serve the React application's production build.
* **Database:** Your project must implement a database. You learnt about MongoDB in the program so naturally that should be your choice for the database. You are free to use any other database as well. If using MongoDb, avoid setting up a local instance of MongoDb as it is not portable and is difficult to share for evaluation and grading. Instead, use MongoDb Atlas service as demonstrated in the Express videos on PRISM. You can use the free tier provided by the Atlas service.
* **Authentication:** Almost every web application features authentication and authorization capability to enable users to sign up and utilise services behind a user account. Implement this capability to bring practical usability to your application. You may go through the authentication process shown in the Express videos on PRISM as a guide.

Lastly, deploy your project on Heroku, Netlify or any other cloud provider's free tier as a proof of concept.

**Organizing Assets**

Before you even begin working on your project, keep the following points in mind to ensure you organise your assets in a way that makes it easy to submit your project for evaluation.

**During Development** : In this phase, organise your code in the following folder structure.

**project-folder :** This should reflect the name of your project and must be in all lower-case, using kebab casing with no space in the name.

1. **frontend :** This folder should contain your React frontend
2. **backend :** This folder should contain your Express based API application, the backend that powers the frontend.
3. **Readme.md :** This markdown file should have instructions on setting up the application for preview and evaluation purposes. Please be extremely thorough with these details as documentation is also part of the evaluation criteria. If you're not sure how to author a markdown file, please watch this: <https://youtu.be/7gVG49fWHFQ>

**Deployment:** When you setup your project for deployment, you will need to follow the steps presented below:

1. Produce a build edition of your React app by running **npm run build**
2. Copy the build edition of the React app in your **server**'s static assets folder so it can be served on the root route when you run your server application.
3. Double check to ensure if this is working as expected.
4. Deploy to the cloud provider of your choice.

**Submitting the Project**

Before submitting the project, please ensure the following points:

* Is the application working correctly?
* Is application code structured as advised?
* Is the app deployed?
* Is the organised code stored in a GitHub repository?
* Does the project include a Readme.md with complete instructions on setting up the application.

If the answer to all the above questions is a Yes, then go ahead and submit the following to your learning advisor:

1. **The GitHub Repo URL**
2. **A URL to the deployed application**

**What should I build?**

Think of your project as a reflection of your skills and abilities. You are not building this project just to complete this program and get feedback; you are building it to showcase your abilities as a software developer.

A good project works to impress potential employers and scouts and naturally works in your favour, so spend time on the project wisely and with great attention to detail.

In this guide, we've attempted to share some ideas with you, however you're free to bring your own ideas to the table and built them out as projects. And while you may be tempted to watch a tutorial or two on the web and build the project, we strongly advise against it, not only because of plagiarism, but because it is akin to cheating yourself and your abilities.

On the pages that follow are some application ideas and problem statements that you may pick from, to build your capstone project. Alternatively, we invite you to bring your own ideas to the table.

**Project Evaluation Criteria**

Upon submission, your project will be meticulously evaluated by our panel of experts against a comprehensive set of criteria. Your project’s effectiveness will be measured in the following domains, with each criterion carrying a maximum of 10 points:

1. **Design (UI):** The visual appeal of your application is paramount. This includes an assessment of the user interface design, encompassing typography, colour palette, iconography, as well as the application’s responsiveness, uniformity, and creative elements. An elegant design enhances user engagement and doesn’t necessarily require complexity.
2. **Functionality**: We assess whether the project aligns with the stipulated requirements and functionalities. Key aspects such as feature completeness, usability, error management, performance, and functionality will be scrutinized.
3. **Code Quality**: The quality of your code is indicative of your craftsmanship. We will evaluate the organization, readability, and adherence to coding best practices.
4. **User Experience (UX):** The project’s user experience will be judged on its intuitiveness and the satisfaction it provides to the end-user. A seamless UX is indicative of a project’s success.

**Scoring**

Projects scoring from 0 to 4 points in any criterion will be considered ‘Below Expectations’, while scores from 5 to 10 points will reflect that the project is ‘Meeting Expectations’.

**Plagiarism Policy**

All projects will undergo a rigorous plagiarism check. If substantial similarity to existing works is found, the project may be subject to disqualification. In such cases, you will be asked to revise and resubmit your project for re-evaluation.

**Project Title: Bookshelf**

**Project Brief**

Bookshelf is a mini social network dedicated to book lovers and enables a place for readers to curate their favourite book titles, add ratings, reviews, and comments on reviews by friends. You may refer to [www.goodreads.com](http://www.goodreads.com) for visual reference on the overall scope of this project. A feature wish list is presented below.

**Feature Set**

The web-based application should provide the following feature set:

* **User Accounts**: Allow visitors to sign up and login using their e-mail IDs and password for simplicity. Adding social logins would be an added plus.
* **My Bookshelf:** This section should allow users to search and add books using a third-party API such as Google Books or NY Times Books API. Once added, users should see a table with the thumbnail cover image, title, author, average rating (if already in the database and rated by other users), a rating meter to allow users to rate the title and the ability to add a text review (with edit/removal capabilities).
* **Dashboard:** When a user logs in, they're taken to the dashboard which should feature the following elements:
  + **Currently Reading:** From the bookshelf, the user should be able to click a button to mark a book as 'currently reading'. This should be displayed on the dashboard along with buttons to update progress which should capture a percentage value and a comment. Also provide, within this section a button called "I've finished" which should mark the progress as 100% and remove the book from the 'currently reading' section on the dashboard and the removal of this tag in the My Bookshelf.   
      
    Graphical user interface, text, application, chat or text message

    Description automatically generated

Ref: Goodreads 1

* + **Social Cards:** As users post updates/reviews/ratings, it should get published publicly and visible on the user's dashboard if the post author is a friend (friends list). An example is shown below, and leeway can be taken in utilising this layout for other things. Social cards should be used to convey reviews and ratings (whenever a user updates) to keep the app simple. A user should only see these for members of their friends list.  
      
    Graphical user interface, text, application

    Description automatically generated

**Project Title: Flixxit**

**Project Brief**

Flixxit aims to be a web application with the likeness and basic feature set of OTT platforms such as Netflix, Prime Video and AppleTV+. While the actual products are built up of a complex web of microservices and infrastructure, the objective here is to build core user functionality.

**Feature Set**

The web-based application should provide the following feature set. Feel free to extend this to make the project more vibrant:

* **User Accounts**: Allow visitors to sign up and login using their e-mail IDs and password for simplicity.
* **User profile:** Provide a user’s profile showing account information, the content they have consumed, and suggestions based on their interest. Enable them to update their preferences.
* **Dashboard:** Allows users to browse titles using horizontally scrollable carousels spread by categories, genres etc.
* **Title View:** Clicking on a title leads to this section where synopsis, rating, and other details of a chosen title may be seen.
* **Search:** Allow users to search various types of content like web series, movies, short films, documentaries etc. You may use a third-party API such as <https://www.themoviedb.org/> for obtaining this data.
* **Watchlist:** Allow users to add programs that they wish to watch later with “Autoplay” feature.
* **Rating:** Allow users to rate any program in form of upvote or downvote and show the count.
* **Video player:** Allow user an option to preview or play the selected content on the platform with “Skip Intro” feature. Add few videos to the app as content, which can be played on the video player. Add at least two video quality (HD, Auto).
* **Payment and subscription:** Allow users to subscribe for monthly or yearly plans for accessing all the features of the app. You can build the feature to the extent of displaying the invoice with a Pay Now button. Further integration with a payment gateway is not required however you may feel free to implement the same using a service like Stripe.
* **About us:** Add information about the features, origin, copyrights, terms, and conditions and help desk details in this section.

**Project Title: ScripVault**

**Project Brief**

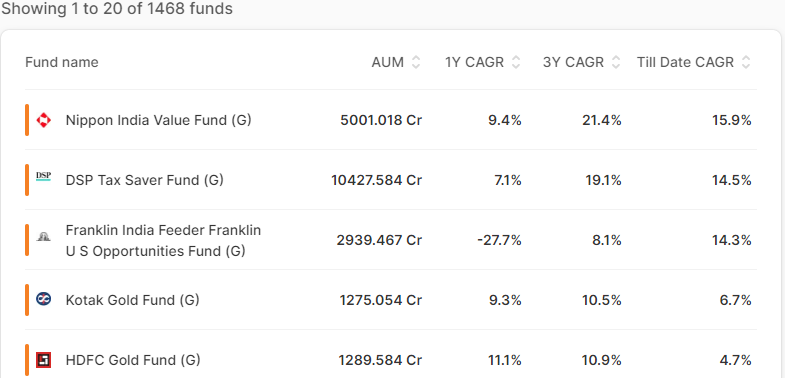
ScripVault is a stock market and mutual fund advisory, investment and tracking platform that allows users to setup goal-oriented investment portfolios and track their progress every day. You may refer to [Scripbox: Invest in Mutual Funds, Fixed Deposit and Financial Planning](https://scripbox.com/) or [Invezta: Best online mutual fund investment platform](https://www.invezta.com/) for visual reference on the overall scope of this project.

Analyse similar platforms for reference and try and build simulated functionality such as the ability to browse mutual funds, place an order for purchase, viewing portfolio etc. A feature wish list is presented below.

**Feature Set**

The web-based application should provide the following feature set:

* **User Accounts**: Allow visitors to sign up and login using their e-mail IDs and password.
* **User Dashboard**: The user dashboard should display:
  + **The user's net worth based on the investments held** - This could be an arbitrary figure from the API
  + **A portfolio**: This would be a list of mutual funds and stocks (categorized) held by the user
  + **Invest**: This button should lead to a list of mutual funds allowing users to pick one fund > set frequency as either one-time investment or Systematic-Investment Plan (SIP) > Set amounts based on the previous choice > Confirm. This should add the investment to the portfolio list.
  + **Profile**: Allow users to update their phone number, password and postal address details only.
* **Explore:** This section should allow users to search and add scrips/stocks. Once added, users should see the name of the script its market value and invested value on their dashboard. This would require an API to be created which produces a fake dataset.
* **Options on home page:** For selection of Mutual funds, fixed deposits, ETFs, NFOs, NPS.
* **Real-time streaming of Charts and Quotes, 1 yr, 3 yr and 5 yr returns:** while exploring the stocks, provide the chart of that stock with two buttons Buy or Sell. Use WebSockets or SSE to implement this using an API that randomly updates prices. **Sample as shown below:**



* **Watchlist:** From the ‘EXPLORE’, the user should be able to click a button to mark a script as 'watchlist'. This should be displayed on the dashboard.
* **Dashboard:** Dashboard should feature the following elements:
  + **Market value:** Market value of the stock purchased
  + **Invested value:** Invested value of the stock purchased
* **Ask Experts:** Suggest portfolios for investments based on the user’s needs. Such as long term, short term, etc. Queries logged here may simply be viewed using a backend dashboard for ease of development.

**Project Title: WeThePeople**

**Project Outline**

WeThePeople is a full-stack crowdfunding web application similar to Kickstarter and built using the MERN stack. Users will have the ability to create and manage crowdfunding campaigns while potential backers can browse, search, and contribute to projects of their choice.

Key components of the front-end may include a Navbar for global navigation, a Home page featuring highlighted projects, a Browse page for searching and filtering projects, a ProjectDetails page to display detailed information and backer tiers, a CreateProject page for project submission, and a UserProfile page for user account management. State management will be handled using Redux or the Context API.

The API will consist of routes to handle user authentication (sign up, log in, log out, password reset), user profile CRUD operations, project CRUD operations (including project updates), and contribution management (including backer tiers). Middleware will be utilized for tasks such as authentication, error handling, and input validation.

**Feature Set**The web-based application should provide the following feature set. Feel free to extend this to make the project more vibrant:

* **User Authentication**: This feature handles user registration, login, logout, and password reset functionality.
* **Project Creation**: This feature allows users to create, edit, and delete their own crowdfunding projects.
* **Project Browsing**: This feature enables users to search, filter, and browse through active projects on the platform.
* **Project Details**: This feature displays detailed information about a project, including description, funding goal, duration, updates, and backer tiers.
* **Contribution Management**: This feature allows backers to select a reward tier, contribute, and view their past contributions.
* **User Profiles**: This feature enables users to view and manage their personal information, crowdfunding projects, and project contributions.
* **Project Updates**: This feature allows project creators to post updates about their project's progress, which can be viewed by backers.
* **Funding Progress**: This feature tracks and displays the funding progress of a project, including the amount raised and the percentage of the funding goal achieved.
* **Featured Projects**: This feature highlights a selection of top or trending projects on the home page, making them more visible to potential backers.
* **Backer Rewards**: This feature allows project creators to set up reward tiers with various perks to incentivize backers to contribute to their project.

**Project Title: SchoolCool**

**Project Outline**

SchoolCool is an all-in-one school management web application that aims to streamline administrative processes, improve communication, and enhance the overall educational experience for students, teachers, and parents. The application will be built using a modern technology stack to provide a user-friendly, efficient, and secure platform.

The front-end will be designed with a responsive and intuitive user interface, ensuring a seamless experience across various devices. The back-end will consist of a robust API that handles all data transactions and supports role-based access control to protect sensitive information. The database will store and manage all relevant data, including student records, staff information, course schedules, attendance, and grades.

**Feature Set**The web-based application should provide the following feature set. Feel free to extend this to make the project more vibrant:

* **User Authentication:** This feature supports the registration, login, and role-based access control for different user types, including administrators, teachers, students, and parents.
* **Student Management:** This feature allows for the creation, modification, and deletion of student records, including personal information, enrolment status, and academic history.
* **Staff Management:** This feature enables the administration to manage staff information, including personal details, employment status, and teaching assignments.
* **Course Scheduling**: This feature facilitates the creation and management of course schedules, including class timings, room assignments, and instructor allocation.
* **Attendance Tracking:** This feature allows teachers to record and track student attendance for each class, while administrators and parents can monitor overall attendance trends.
* **Grade Management:** This feature enables teachers to input and manage student grades, as well as generate report cards and progress reports for parents.
* **Parent Portal:** This feature provides parents with access to their children's academic information, including attendance, grades, and teacher feedback.
* **Communication Tools**: This feature offers messaging functionality for direct communication between teachers, students, and parents, as well as group messaging for announcements and notifications.
* **Resource Management**: This feature allows administrators to manage school resources, such as classrooms, labs, and equipment, and track their usage and availability.
* **Dashboard and Reporting**: This feature provides users with personalized dashboards and generates various reports to help monitor performance and make data-driven decisions.

**Project Title: KnowledgeCube**

**Project Outline**

KnowledgeCube is an online learning platform designed to empower individuals and organizations to create and share short courses on various subjects. The platform will be built using a modern technology stack to ensure a seamless and engaging user experience for both course creators and learners. Key components of the platform will include a course creation suite, a content delivery system, and a user management system.

The front-end will be designed with a responsive and intuitive user interface, ensuring a seamless experience across various devices. The back-end will consist of a robust API that handles all data transactions and supports role-based access control to protect sensitive information. The database will store and manage all relevant data, including user information, course details, and learning progress.

**Feature Set**The web-based application should provide the following feature set. Feel free to extend this to make the project more vibrant:

* **User Authentication:** This feature supports the registration, login, and role-based access control for different user types, such as administrators, course creators, and learners.
* **Course Creation Suite**: This feature enables course creators to design, publish, and manage short courses, including setting up course structure, uploading content, and creating quizzes or assessments.
* **Content Management**: This feature allows for the storage, organization, and retrieval of various types of course content, such as text, images, videos, and documents.
* **Course Discovery**: This feature enables users to search, filter, and browse through available courses based on various criteria, such as subject, difficulty level, or course creator.
* **Enrolment and Progress Tracking**: This feature allows learners to enrol in courses, track their learning progress, and resume learning where they left off.
* **Interactive Learning Experience**: This feature supports interactive elements, such as quizzes, assessments, and gamification elements, to make the learning process more engaging and effective.
* **Learner Dashboard**: This feature provides learners with a personalized dashboard displaying their enrolled courses, progress, and achievements.
* **Course Creator Dashboard**: This feature offers course creators insights into course performance, learner engagement, and feedback to help improve course quality.
* **Social Features**: This feature allows users to interact with each other through comments, discussion boards, and direct messaging, promoting a sense of community and collaborative learning.
* **Certificates and Badges**: This feature enables the issuance of completion certificates and achievement badges to learners who successfully complete courses or reach specific milestones.

**Project Title: Black**

**Project Outline**

Black is a Discord-like web application that offers a platform for real-time communication and collaboration among users. The application will be built using a modern technology stack to deliver a seamless and engaging user experience. Key components of the platform will include chat rooms, voice and video calls, file sharing, and user management.

The front-end will be designed with a responsive and intuitive user interface, ensuring a seamless experience across various devices. The back-end will consist of an API that handles all data transactions and supports role-based access control to protect sensitive information. The database will store and manage all relevant data, including user information, chat history, and file attachments.

**Feature Set**The web-based application should provide the following feature set. Feel free to extend this to make the project more vibrant:

* **User Authentication:** This feature supports the registration, login, and role-based access control for different user types, such as administrators and members.
* **Chat Rooms**: This feature enables users to create, join, and manage public and private chat rooms for text-based communication.
* **Voice and Video Calls:** This feature allows users to initiate and participate in voice and video calls within chat rooms or in private conversations.
* **File Sharing**: This feature supports the sharing of files, such as images, documents, and videos, within chat rooms or in private conversations.
* **Direct Messaging**: This feature enables users to send private text messages, voice messages, and file attachments to other users.
* **User Management**: This feature allows administrators to manage user accounts, including assigning roles, banning users, and managing permissions.
* **Notifications:** This feature provides users with real-time notifications for new messages, mentions, and other important events.
* **Customization**: This feature enables users to customize their profiles, set avatars, and choose themes for the application interface.
* **Search and Filtering**: This feature allows users to search for specific messages, users, or chat rooms based on keywords or other criteria.
* **Integration**: This feature supports integration with third-party applications and services, such as calendar apps, project management tools, and more.

**Project Title: BankFresh**

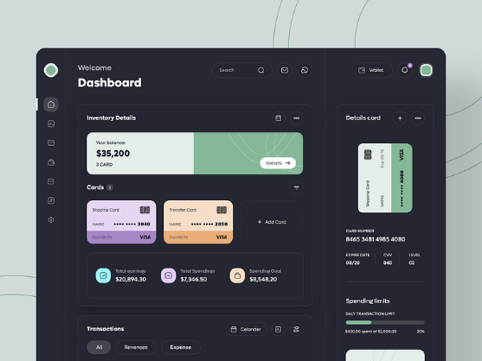
**Project Outline**

BankFresh is a dynamic net banking web application designed to provide users with a secure and smooth banking experience. It offers key banking features such as account access, transaction history, fund transfers, and digital statement downloads. You may refer to any online banking application for visual reference on the overall scope of this project. A feature wish list is presented below.

**Feature Set**

The web-based application should provide the following feature set:

* **User Accounts**: Allow users to register and login using their e-mail IDs and password for simplicity.
* **User Authentication and Security:** The application should provide the following features with respect to security:
  + Multi-factor authentication (MFA) for an additional layer of security.
  + Password reset and recovery mechanisms.
* **Dashboard:** When a user logs in, they're taken to the dashboard which should feature the following elements:
  + **Account Overview:** This section should display a summary of the user's account balances, including savings, checking, credit cards, loans, and investments.
  + **Transaction History:** It should show a list of recent transactions for all linked accounts.
  + **Quick Actions:** This section should offer shortcuts for common actions like fund transfers, pay bills and account management. Users should be able to initiate transactions directly from the dashboard.



*Ref: Dribbble*

* **Fund Transfers:** Users should be able to perform the following activities,
  + Add a new payee.
  + Perform transfers between own accounts.
  + Perform transfers to other accounts within the same bank.
  + Transfer funds to external accounts at other banks.
* **Digital Statements:** Users should be able to download digital account statements and reports within selected date range in PDF format.
* **Bill Payments:** Users should be able to pay utility bills, credit card bills, loans, and other payments online.
* **Credit Card Services:** Users should be able to
  + View credit card statements and transaction history.
  + Make credit card payments and manage rewards.

**Project Title: The Gram**

**Project Overview**

TheGram is a modern web-based social media application that aims to offer users a dynamic platform for sharing photos, following friends, liking, and commenting on posts, and private messaging. You may refer to Instagram for visual reference on the overall scope of this project. A feature wish list is presented below.

**Feature Set**

The web-based application should provide the following feature set:

* **User Accounts**: Allow visitors to sign up and login using their e-mail IDs and password for simplicity. Adding social logins would be an added plus. Once registered, users should be able to change or reset their password.
* **User Profile:** This section should allow users to view their own profile which includes the following.
  + Profile picture
  + Bio
  + Number of follower and & following profiles
  + All the posts in descending order (latest being the first)
* **Change Profile:** Users should be able to update their profile picture and bio
* **Create Post:** Users should be able to upload and share photos by creating a new post. They can add captions, location information, and tags to their posts.
* **Search and Engage:** Users should be able to search for another user profile, follow and unfollow them.
* **Interact:** Users should be able to like/unlike and comment to a post.
* **Private Messaging:** Users should be able to send messages to another user privately.

**Project Title: ShopPlusPlus**

**Project Overview**

ShopPlusPlus is a dynamic e-commerce web application that offers users an immersive online shopping venture. It provides features such as user account management, Wishlist, product browsing, and secure transactions. You may refer to Amazon for visual reference on the overall scope of this project. A feature wish list is presented below.

**Feature Set**

The web-based application should provide the following feature set:

* **User Account Management**: Allow visitors to sign up and log in using their e-mail IDs/mobile numbers and password for simplicity. Adding social logins would be an added plus. After logging in, users should be provided with the following features:
  + **Edit user information-** Allow users to update name, registered contact number, registered email address and password.
  + **Address book-** Allow users to add, update and delete delivery addresses. Users can add as many addresses as they want.
* **Product Browsing:** This section should allow users to explore and browse through various product categories. Users should be able to filter and search for products based on specific criteria. This feature can also be used by a visitor (without logging in).
* **Wishlist:** This section allows users to add and remove products for future interaction. Users should be able to add the wishlisted products to the cart. Only logged in users should be able to use this feature.
* **Cart:** This section allows users to add and remove products for placing a new order. This section should also display the total bill and update it as and when the user updates the cart. Only logged in users should be able to use this feature. Note that users can directly add a product to the cart without even adding them to the Wishlist.
* **Order History & Tracking:** This section provides the following:
  + Access order history and tracking details
  + Provide timely updates on order processing and delivery stages.

Only logged in users should be able to use this feature.

* **User Reviews and Ratings:** This feature should:
  + Allow users to submit reviews and assign ratings to products they purchased.
  + Display aggregated ratings and user feedback to help in purchasing decisions.
* **Secure Purchase Handling:** This section should provide a robust payment processing system that guarantees secure and seamless transactions.